Parallel Rollout Approximation for Pixel-Space Autoregressive Image Generation

Jiayi Xu[1,2] **Di He**[1] **Guolin Ke**[2*]
[1] Peking University [2] DP Technology

Abstract

Pixel-space continuous-token autoregressive (AR) generation directly models images as sequences of raw pixel patches, avoiding discrete tokenization or a separately pretrained tokenizer. However, it faces coupled challenges: high-dimensional patch generation causes large single-step errors, and teacher-forced training creates a train–inference gap that makes these errors accumulate across AR steps. Existing fixes such as x-prediction and input noise injection only partially mitigate these issues. Exact rollout training better matches inference-time conditions, but is impractical due to prohibitively slow sequential sampling. We propose *Parallel Rollout Approximation* (PRA), a scalable framework that addresses both challenges jointly. PRA generates low-dimensional intermediate states instead of high-dimensional pixel patches, then maps them back to pixel-space tokens with a pixel decoder, preserving a pixel-in, pixel-out AR interface. It also constructs inference-like pixel inputs through the same intermediate-state-to-pixel path used at inference, independently across positions, approximating the pixel-feedback interface encountered during inference-time rollout while retaining parallel teacher-forced training. On class-conditional ImageNet-1K generation at 256×256 resolution, PRA-S with 135M parameters achieves an FID of 2.58, surpassing the previous billion-scale pixel-space AR result of 3.60. Scaling to PRA-L with 511M parameters further improves FID to 1.94, establishing a new state of the art among pixel-space AR models. Beyond generation, PRA achieves higher ImageNet classification probing accuracy than other AR and diffusion baselines, suggesting its potential for unified pixel-space image generation and understanding.

Figure 1: **Left:** FID comparison of pixel-space AR models across parameter scales. PRA achieves substantially lower FID than prior baselines, with PRA-S (135M) already outperforming billion-parameter models. **Right:** Uncurated 256×256 samples generated by PRA-L.

*Corresponding author. Email: kegl@dp.tech
Code: https://github.com/MangataX/PRA

Preprint.

1 Introduction

The success of autoregressive (AR) generation in large language models (LLMs) [Radford et al., 2018, Brown et al., 2020] has motivated extending this paradigm to image generation and other continuous domains [Li et al., 2024, Meng et al., 2024, Teng et al., 2025, Lu et al., 2025]. Most successful AR image generators, however, operate in a discrete or latent token space rather than directly in pixel space. Discrete-token approaches rely on vector quantization or pretrained tokenizers to convert images into codebook indices [Gray, 1984, Van Den Oord et al., 2017, Esser et al., 2021, Yu et al., 2021], while recent continuous-token AR methods typically model continuous tokens in learned latent or feature spaces [Tschannen et al., 2024, Li et al., 2024, Sun et al., 2024b]. Although these token spaces make autoregressive modeling more tractable, they introduce an additional stage and make the final generation quality constrained by the tokenizer or autoencoder. This raises a natural question: can we build a competitive pixel-space autoregressive model trained end-to-end on raw pixel patches, without relying on an external pretrained tokenizer?

Pixel-space autoregressive generation appears conceptually simple: an image can be divided into patches, and the model predicts the next raw pixel patch conditioned on previous ones. In this process, raw pixel patches play a dual role: they are the continuous targets to be generated at the current step, and once generated, they become part of the causal context for future steps. As illustrated in Figure 2, this dual role exposes two coupled challenges. On the output-side, each raw pixel patch is a high-dimensional continuous vector, which is hard to predict in a single AR step and causes large single-step errors. On the input-side, teacher-forced training conditions the AR model on clean ground-truth prefixes, whereas inference requires conditioning on previously generated pixel patches, creating a train–inference mismatch. Together, these challenges form a self-reinforcing mode: difficult high-dimensional predictions introduce large local errors, and the autoregressive loop turns these errors into imperfect future contexts, allowing them to propagate and compound across subsequent steps [Bengio et al., 2015, Ranzato et al., 2016].

Existing techniques improve pixel-space AR baselines but only partially resolve this coupled difficulty. Prediction parameterizations such as x-prediction with the v-loss, which are effective in pixel-space diffusion models [Li and He, 2025], reduce high-dimensional output error but still leave a large gap to diffusion baselines. For the input-side mismatch, injecting noise into ground-truth tokens exposes the AR model to imperfect contexts during training [Bengio et al., 2015, Ke and Xue, 2026, Pasini et al., 2024], but these perturbations are independent of the model's own generation process and therefore cannot fully match the structured, model-dependent errors encountered during inference-time rollout. Naive on-policy rollout would directly expose the model to the inference-time generated prefixes, but it is impractical for continuous-token AR as it requires prohibitively slow sequential autoregressive sampling with multi-step generation [Li et al., 2024] for each token.

We propose Parallel Rollout Approximation (PRA), a scalable framework for pixel-space autoregressive generation that addresses both bottlenecks jointly. As shown in Figure 2, PRA generates compact intermediate states instead of directly predicting high-dimensional pixel patches, reducing single-step generation difficulty. These states are decoded back to pixel tokens at each step, so unlike latent-space AR, the autoregressive backbone still interacts with pixels: it receives pixel prefixes and produces pixel outputs through the decoder at each AR step. To reduce train–inference mismatch, PRA further constructs inference-like training inputs by perturbing intermediate states and passing them through the same intermediate-state-to-pixel path used at inference. As these decoded pixel inputs are built independently across positions, PRA approximates inference-like tokens while retaining parallel teacher-forced training.

We validate PRA on class-conditional ImageNet-1K generation at 256×256 resolution, using raw pixel patches as continuous tokens [Deng et al., 2009]. PRA substantially improves the quality of pixel-space AR generation across model scales: the 135M-parameter PRA-S already surpasses prior billion-scale pixel-space AR models, and PRA-L reaches an FID of 1.94. Moreover, ablations show that low-dimensional intermediate states and decoded pixel inputs contribute complementary gains, supporting our diagnosis of pixel-space AR. The generation-trained backbone also achieves stronger ImageNet linear probing accuracy than AR and diffusion baselines[Ke and Xue, 2026, Li and He, 2025, Peebles and Xie, 2023], suggesting the promise of end-to-end pixel-space autoregressive learning for both generation and visual understanding.

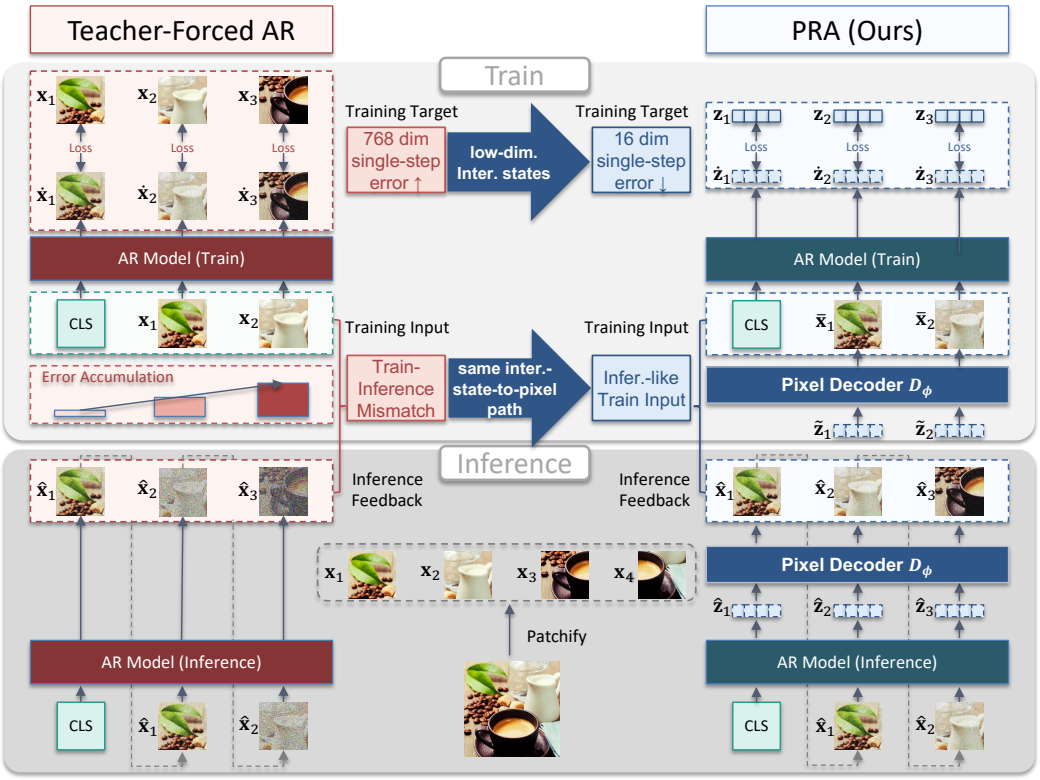

Figure 2: PRA addresses both output-side and input-side challenges in pixel-space AR generation. **Output-side:** Instead of directly predicting high-dimensional pixel patches, PRA generates low-dimensional intermediate states and decodes them back to pixel tokens, reducing single-step generation errors. **Input-side:** Instead of training only on clean ground-truth prefixes, PRA constructs decoded, inference-like pixel inputs in parallel through the same pixel decoder used at inference, reducing train–inference mismatch. This yields a rollout-like pixel-space generated prefix without sequential rollout, while preserving a pixel-in, pixel-out AR interface.

2 Related Work

Train–Inference Mismatch and Rollout-Based Training. Teacher-forced autoregressive training creates a mismatch between training and inference: models are trained on ground-truth prefixes but must condition on their own generated prefixes at inference time, leading to exposure bias and error accumulation [Bengio et al., 2015, Ranzato et al., 2016, Schmidt, 2019]. Existing remedies expose models to generated or perturbed inputs during training, including scheduled sampling, sequence-level rollout training, adversarial dynamics matching, and recent on-policy distillation [Bengio et al., 2015, Ranzato et al., 2016, Goyal et al., 2016, Agarwal et al., 2024]. However, exact rollout-based training is inherently sequential and becomes especially expensive for continuous-token AR models, where each token may require multi-step diffusion sampling. In contrast, PRA approximates the benefit of rollout training by constructing inference-like pixel inputs in parallel, avoiding sequential autoregressive rollouts during training.

Continuous-Token Autoregressive Generation. Continuous-token AR models extend autoregressive generation from discrete vocabularies to continuous-valued tokens, such as latent features or raw signal patches. Recent methods commonly use token-level diffusion heads to model continuous next-token distributions [Li et al., 2024]. Many strong continuous-token AR methods operate in latent space: they rely on a pretrained tokenizer or autoencoder to map raw signals into lower-dimensional continuous tokens, and then train the AR model on these latent sequences. Prior work further shows that such latent spaces must be carefully designed for stable AR generation, using techniques such as stronger noise injection [Sun et al., 2024b, Pasini et al., 2024] or constant-norm latent representations [Ke and Xue, 2026, Team et al., 2025]. These designs improve robustness by making continuous-token generation easier or less sensitive to accumulated errors. PRA instead targets

pixel-space continuous-token AR: it learns low-dimensional intermediate states end-to-end with the AR model, without a separately pretrained tokenizer, while keeping the external interface pixel-in and pixel-out.

Pixel-Space Generative Modeling. Pixel-space image generation has been dominated by diffusion models, which generate images through iterative denoising directly in the original pixel space [Li and He, 2025, Wang et al., 2025a, Chen et al., 2025]. Pixel-space diffusion models also handle high-dimensional pixel outputs, but they do not feed locally generated patches back into a causal context across spatial positions. Pixel-space autoregressive generation, in contrast, remains relatively underexplored and has generally underperformed diffusion-based models [Tschannen et al., 2025, Li et al., 2025, Zheng et al., 2025b]. A key bottleneck is that each generated high-dimensional patch becomes part of the context for later predictions, making AR models vulnerable to cross-step error accumulation. PRA improves the viability of pixel-space AR by reducing per-step generation difficulty with low-dimensional intermediate states and by constructing inference-like pixel inputs in parallel to mitigate train–inference mismatch.

3 Pixel-Space Image Generation via Continuous-Token AR Modeling

Rather than generating images inefficiently at the pixel level, we adopt a patch-wise formulation [Li and He, 2025], where all pixels within each patch are grouped into a single continuous token. For example, with a patch size of 16^2, each token corresponds to a $16 \times 16 \times 3 = 768$-dimensional continuous vector. Given an image, we arrange its patch-wise continuous tokens in raster-scan order as $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$ with $\mathbf{x}_i \in \mathbb{R}^d$. Under this formulation, pixel-space autoregressive image generation becomes a continuous-token autoregressive generation problem, with the joint distribution factorized as $p(\mathbf{x}) = \prod_{i=1}^{T} p(\mathbf{x}_i \mid \mathbf{x}_{<i})$.

We use $\mathbf{x}_i$ to denote a ground-truth continuous token and $\hat{\mathbf{x}}_i$ to denote a model-generated continuous token. A causal Transformer encodes the prefix tokens. Under teacher forcing, the model takes the ground-truth prefix $\mathbf{x}_{<i}$ as input and produces a hidden state $\mathbf{h}_{i-1} = f_\theta(\mathbf{x}_{<i})$, where θ denotes the Transformer parameters. Optional conditioning information, such as class labels or text prompts, can be prepended to the token sequence and included in the causal context.

To model the continuous next-token distribution, we attach a token-level diffusion head to the AR Transformer [Li et al., 2024]. Conditioned on $\mathbf{h}_{i-1}$, the head learns to transform a simple prior sample into the target token $\mathbf{x}_i$. We train this head with a rectified-flow objective [Liu et al., 2022]. Given a prior sample $\mathbf{x}_i^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the target token $\mathbf{x}_i^1 = \mathbf{x}_i$, and a time variable $t \sim \mathcal{U}(0, 1)$, we form the interpolated state $\mathbf{x}_i^t = (1-t)\mathbf{x}_i^0 + t\mathbf{x}_i^1$. The diffusion head v_ω takes $(\mathbf{x}_i^t, t, \mathbf{h}_{i-1})$ as input and is trained to predict either the straight-path velocity $\mathbf{x}_i^1 - \mathbf{x}_i^0$ or the clean target $\mathbf{x}_i^1$.

At inference time, the model conditions on previously generated tokens $\hat{\mathbf{x}}_{<i}$ and computes $\hat{\mathbf{h}}_{i-1} = f_\theta(\hat{\mathbf{x}}_{<i})$. The diffusion head samples the next token by starting from Gaussian noise and integrating the learned velocity field from $t = 0$ to $t = 1$ with N Euler-Maruyama steps. The final state is taken as the generated continuous token $\hat{\mathbf{x}}_i$, which is then fed back as input for subsequent autoregressive steps. This differs from teacher-forced training, where the model conditions on ground-truth prefixes $\mathbf{x}_{<i}$, creating a train–inference gap that can cause errors to accumulate during generation.

4 Challenges of Pixel-Space Autoregressive Generation

Although the continuous-token AR formulation is conceptually straightforward, we find that naive pixel-space AR degrades substantially at higher resolutions. To systematically diagnose this degradation, we compare against JiT [Li and He, 2025], a pixel-space diffusion model, under two controlled settings: 64^2 resolution with a 4^2 patch size, and 256^2 resolution with a 16^2 patch size. These settings have the same autoregressive length, $(64/4)^2 = (256/16)^2 = 256$, but very different token dimensionalities, $4^2 \times 3 = 48$ versus $16^2 \times 3 = 768$. For fairness, the AR model and JiT use the same model scale (130M parameters) and the same training budget (200 epochs) on ImageNet [Deng et al., 2009]. We report FID computed on 50k generated samples, with classifier-free guidance scales tuned for each model. We then introduce diagnostic variants to isolate the two challenges: output-side token dimensionality and input-side train–inference mismatch, with results summarized in Table 1.

Output-side challenge: high-dimensional tokens increase single-step errors. As shown in Table 1, the AR model is competitive with JiT when the token dimensionality is low, but falls much further behind when the token dimensionality increases from 48 to 768. Since both settings use the

Table 1: Diagnostics of pixel-space AR generation. All settings use 256 autoregressive tokens. Res/Patch denotes resolution/patch size, Noise denotes input noise injection, Tok. Dim. denotes token dimensionality. Output-side diagnostics compare JiT and AR under different token dimensionalities. Input-side diagnostics compare AR runs with and without input noise injection.

Study	ID	Res/Patch	Model	Pred.	Noise	Tok. Dim.	Epoch	FID ↓
Output	D1	64/4	Diffusion (JiT)	x	–	48	200	3.55
	A1	64/4	AR	x	✓	48	200	4.06
	A2	64/4	AR	v	✓	48	200	4.15
	D2	256/16	Diffusion (JiT)	x	–	768	200	4.56
	A3	256/16	AR	x	✓	768	200	7.68
	A4	256/16	AR	v	✓	768	200	9.70
Input	A5	64/4	AR	x	✗	48	200	6.71
	A6	256/16	AR	x	✗	768	200	9.94

same number of autoregressive steps, this degradation cannot be explained by a longer generation horizon. Instead, it points to high-dimensional continuous-token generation as a key output-side bottleneck for pixel-space AR. We further compare the high-dimensional AR setting with and without the x-prediction variant from JiT. Similar to its effect in diffusion models, x-prediction improves AR performance from 9.70 to 7.68 compared with v-prediction (A3 vs. A4). However, the gap to JiT remains large, suggesting that this output-side fix alone is insufficient: directly generating high-dimensional pixel tokens still causes substantial single-step errors.

Input-side challenge: train–inference mismatch amplifies errors across steps. The input-side difficulty comes from the mismatch between training on ground-truth prefixes and inference on generated prefixes. Once a generated token is fed back into the autoregressive context, its error can influence later predictions and accumulate across steps. We therefore strengthen the AR baseline with input noise injection: each ground-truth input token is perturbed along the rectified-flow interpolation used by the prediction head, $\tilde{\mathbf{x}}_i^t = (1-t)\mathbf{x}_i^0 + t\mathbf{x}_i$, where $\mathbf{x}_i^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $t \sim \mathcal{U}(t_{\min}, 1)$, before being fed into the causal context. This improves FID from 9.94 to 7.68 (A6 to A3), but the gap to JiT remains substantial, indicating that independent noise perturbations cannot fully match the structured, model-dependent errors produced during inference-time rollout.

These diagnostics show that pixel-space AR requires addressing both sides of the problem: reducing the difficulty of generating high-dimensional pixel tokens, and training the model under inputs that better resemble the generated tokens it receives during inference-time rollout. PRA tackles these two issues jointly by generating compact intermediate states and constructing decoded inference-like inputs in parallel.

5 Parallel Rollout Approximation

The diagnostics in Sec. 4 reveal two coupled challenges in pixel-space AR generation. On the output side, directly generating high-dimensional pixel patches leads to large single-step errors. On the input side, teacher-forced training exposes the model to clean ground-truth prefixes, whereas inference requires conditioning on generated pixel tokens, causing a train–inference gap and error accumulation. Although x-prediction and input noise injection mitigate these two issues respectively, the resulting AR model still lags substantially behind pixel-space diffusion models.

We address these challenges with *Parallel Rollout Approximation* (PRA). On the output side, PRA generates low-dimensional intermediate states instead of high-dimensional pixel patches, and decodes these states back to pixel-space tokens so that each AR step still produces a pixel patch. On the input side, PRA uses the same pixel decoder to construct inference-like pixel inputs during training, approximating the pixel-space generated tokens during inference-time rollout without executing costly sequential autoregressive sampling.

5.1 End-to-End Intermediate Targets for Pixel Outputs

PRA first addresses the output-side difficulty of directly generating high-dimensional pixel patches by introducing an end-to-end learned intermediate target. As illustrated in Figure 2, for each ground-truth

pixel token $\mathbf{x}_i \in \mathbb{R}^d$, we construct a low-dimensional state $\mathbf{z}_i \in \mathbb{R}^{d_z}$ with $d_z < d$, and use a causal pixel decoder D_ϕ to map it back to the pixel-token space. The decoder is causal and trained to reconstruct the original token, i.e., $\mathbf{x}_i^{\text{rec}} = D_\phi(\mathbf{z}_i, \mathbf{z}_{<i}) \approx \mathbf{x}_i$. This reconstruction path constrains $\mathbf{z}_i$ to retain the information needed for pixel-space output, while allowing the AR model to generate in a lower-dimensional space.

A straightforward way to obtain such low-dimensional targets is to use an external VAE or tokenizer. However, this would introduce a separate representation-learning stage and turn the model into a two-stage latent AR system. We instead learn the intermediate targets end-to-end with the AR model. The simplest end-to-end construction is to map the current ground-truth token alone, e.g., $\mathbf{z}_i = g_\psi(\mathbf{x}_i)$. However, such a target is purely local and ignores the causal prefix representation already computed by the AR Transformer.

PRA therefore constructs context-aware intermediate targets. Under teacher forcing, the AR Transformer encodes the ground-truth prefix and produces the causal representation $\mathbf{h}_{i-1} = f_\theta(\mathbf{x}_{<i})$. We combine this prefix representation with the current ground-truth token through a lightweight projection network:

$$\mathbf{z}_i = g_\psi(\mathbf{x}_i, \mathbf{h}_{i-1}), \quad \mathbf{z}_i \in \mathbb{R}^{d_z}, \quad d_z < d. \tag{1}$$

Here, $\mathbf{x}_i$ is used only to define the training target $\mathbf{z}_i$; the AR model itself must generate this state from the causal prefix. Using $\mathbf{h}_{i-1}$ introduces little additional cost, since the prefix representation is already available from the teacher-forced AR forward. To prevent g_ψ from relying only on the current token, we apply token masking: with probability p_{mask}, $\mathbf{x}_i$ is replaced by a learnable mask embedding before projection, encouraging $\mathbf{z}_i$ to incorporate causal context.

Given these intermediate targets, the token-level diffusion head is trained in the low-dimensional space using $\mathbf{z}_i$ as the target state. During generation, the AR model generates an intermediate state $\hat{\mathbf{z}}_i$, and the pixel decoder maps it to the pixel-space token $\hat{\mathbf{x}}_i = D_\phi(\hat{\mathbf{z}}_i, \hat{\mathbf{z}}_{<i})$. Thus, PRA reduces the difficulty of each AR generation step while preserving a pixel-in, pixel-out interface.

5.2 Parallel Construction of Inference-Like Pixel Inputs

Low-dimensional intermediate targets reduce the output-side difficulty of generating each token, but they do not by themselves resolve the input-side train–inference gap. If the AR model is still trained only on clean ground-truth pixel prefixes, it may remain brittle when conditioned on decoded tokens produced by its own generations during inference-time rollout. PRA therefore approximates this rollout pixel-input interface in parallel. Rather than sampling a full autoregressive trajectory, it constructs decoded pixel inputs independently at each position through the same intermediate-state-to-pixel path used at inference.

For each target intermediate state $\mathbf{z}_i$, we sample $\mathbf{z}_i^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $t \sim \mathcal{U}(t_{\min}, 1)$, and form a perturbed state

$$\tilde{\mathbf{z}}_i^t = (1-t)\mathbf{z}_i^0 + t\mathbf{z}_i. \tag{2}$$

The causal pixel decoder then maps this perturbed state back to pixel space:

$$\bar{\mathbf{x}}_i = D_\phi(\tilde{\mathbf{z}}_i^t, \tilde{\mathbf{z}}_{<i}^t). \tag{3}$$

The resulting token $\bar{\mathbf{x}}_i$ is not sampled from an actual autoregressive trajectory. Instead, it serves as an inference-like pixel input because it is produced through the same intermediate-state-to-pixel decoding path used at inference time. This makes it closer to inference-time decoded tokens than either clean ground-truth inputs or independent pixel-space noise injection.

The reconstructed sequence $\bar{\mathbf{x}} = (\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_T)$ is then used as a stop-gradient input sequence for AR training. With the standard causal shift, the hidden state for position i is computed from the reconstructed prefix $\bar{\mathbf{x}}_{<i}$ rather than the clean prefix $\mathbf{x}_{<i}$, and the token-level diffusion head is trained to generate the target intermediate state $\mathbf{z}_i$. Since all $\bar{\mathbf{x}}_i$ are constructed independently from their corresponding target states, this input construction remains fully parallel over positions. In practice, PRA requires an additional parallel AR forward during training, but avoids the sequential sampling cost.

5.3 Overall Training and Inference Pipeline

Alg. 1 and 2 summarizes the overall training and inference pipeline of PRA. During training, PRA first runs a teacher-forced AR forward on the ground-truth pixel sequence to obtain the causal

Algorithm 1 PRA training

Require: image IMG, pixel tokens $\mathbf{x}_{1:T}$, condition c
Require: AR Transformer f_θ, target encoder g_ψ, causal pixel decoder D_ϕ, flow head v_ω

for $i = 1, \dots, T$ **in parallel do**
  $\mathbf{h}_{i-1} \leftarrow f_\theta(\mathbf{x}_{<i})$
  $\mathbf{z}_i \leftarrow g_\psi(\mathbf{x}_i, \mathbf{h}_{i-1})$ ▷ low-dim inter. target
  sample $\mathbf{z}_i^0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $t \sim \mathcal{U}(t_{\min}, 1)$
  $\tilde{\mathbf{z}}_i^t \leftarrow (1-t)\mathbf{z}_i^0 + t\mathbf{z}_i$
  $\bar{\mathbf{x}}_i \leftarrow D_\phi(\tilde{\mathbf{z}}_i^t, \tilde{\mathbf{z}}_{<i}^t)$ ▷ decoded rollout-like pixel token
end for

for $i = 1, \dots, T$ **in parallel do**
  $\bar{\mathbf{h}}_{i-1} \leftarrow f_\theta(c, \mathrm{sg}(\bar{\mathbf{x}}_{<i}))$
  sample $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $s_i \sim \mathcal{U}(0, 1)$
  $\mathbf{z}_i^{s_i} \leftarrow (1-s_i)\epsilon_i + s_i\mathbf{z}_i$ ▷ Flow Matching train
end for
$\mathcal{L}_{\mathrm{AR}} \leftarrow \frac{1}{T}\sum_{i=1}^{T} \left\| v_\omega(\mathbf{z}_i^i, s_i, \bar{\mathbf{h}}_{i-1}) - (\mathbf{z}_i - \epsilon_i) \right\|_2^2$
$\mathcal{L}_{\mathrm{rec}} \leftarrow \ell_{\mathrm{rec}}(\mathrm{Unpatchify}(\bar{\mathbf{x}}_{1:T}), \mathtt{IMG})$
$\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{AR}} + \mathcal{L}_{\mathrm{rec}} + \mathcal{L}_{\mathrm{aux}}$
return $\mathcal{L}$

Algorithm 2 PRA inference: sequential rollout

Require: condition c
Require: f_θ, D_ϕ, v_ω

$\hat{\mathbf{x}}_{<1} \leftarrow \emptyset$
for $i = 1, \dots, T$ **do**
  $\hat{\mathbf{h}}_{i-1} \leftarrow f_\theta(c, \hat{\mathbf{x}}_{<i})$ ▷ generated prefix
  $\hat{\mathbf{z}}_i \leftarrow \textsc{FlowSample}(v_\omega, \hat{\mathbf{h}}_{i-1})$
  $\hat{\mathbf{x}}_i \leftarrow D_\phi(\hat{\mathbf{z}}_i, \hat{\mathbf{z}}_{<i})$ ▷ decoded rollout pixel token
  append $\hat{\mathbf{x}}_i$ to the prefix
end for
return $\mathrm{Unpatchify}(\hat{\mathbf{x}}_{1:T})$

representations used for constructing the target intermediate states $\mathbf{z}_i$. These target states are then perturbed and passed to the pixel decoder, which reconstructs the corresponding pixel tokens and produces inference-like pixel inputs $\bar{\mathbf{x}}_i$. In a second parallel AR forward, the model conditions on the reconstructed prefix $\bar{\mathbf{x}}_{<i}$ and trains the token-level diffusion head to generate the target state $\mathbf{z}_i$. Thus, the AR model learns to generate intermediate states under imperfect pixel prefixes, while the pixel decoder learns to map perturbed intermediate states back to pixel space.

During inference, PRA uses the same intermediate-state-to-pixel path autoregressively. At each step, the model conditions on previously generated pixel tokens $\hat{\mathbf{x}}_{<i}$, generates a low-dimensional intermediate state $\hat{\mathbf{z}}_i$, decodes it into the next pixel token $\hat{\mathbf{x}}_i = D_\phi(\hat{\mathbf{z}}_i, \hat{\mathbf{z}}_{<i})$, and feeds this token back for subsequent generation. Therefore, PRA keeps the external AR interface pixel-in and pixel-out, while using low-dimensional intermediate states only internally.

5.4 Discussion

Pixel-space AR vs. diffusion models. Pixel-space diffusion models such as JiT also model high-dimensional pixel patches, but they avoid the autoregressive loop that makes local errors accumulate across spatial positions. In pixel-space AR, each generated patch is fed back into the causal context for subsequent predictions, so single-step errors can become non-local and progressively degrade later patches. This makes train–inference mismatch a key bottleneck for pixel-space AR, even when using x-prediction that work well for diffusion models. PRA addresses this issue by constructing rollout-like pixel decoded pixel inputs in parallel, while also reducing single-step difficulty through low-dimensional intermediate states, thereby narrowing the gap to diffusion-based approaches.

PRA vs. two-stage latent AR models. Two-stage latent AR models reduce generation difficulty by relying on an external tokenizer or autoencoder, often with AR-specific regularization of the latent space. PRA also introduces low-dimensional intermediate states, but learns them end-to-end with the AR model and uses them only as internal generation targets. This removes the need for a separately pretrained tokenizer while preserving a pixel-in, pixel-out autoregressive interface: the model takes pixel patches as inputs and generates pixel patches as outputs. In pixel-space modeling, this unified raw-token interface can benefit both image generation and visual understanding, as reflected by PRA's stronger image classification probing performance compared with latent-space AR baselines (Sec. 6.5).

6 Experiments

We conduct a systematic evaluation of PRA on class-conditional ImageNet-1K generation at 256×256 resolution, using raw pixel patches as continuous tokens. We first compare PRA against strong baselines from two-stage latent-space generation, pixel-space diffusion, and pixel-space AR. We then validate the two main components of PRA, end-to-end intermediate targets for pixel outputs and parallel inference-like pixel inputs, through output-side and input-side ablations. Finally, beyond generation quality, we evaluate image understanding through ImageNet classification probing.

6.1 Implementation Details

We evaluate three model scales: PRA-S (135M), PRA-B (250M), and PRA-L (511M). All variants use the same PRA architecture and differ only in model depth and width, with architecture details provided in Appendix A. We use 16×16 pixel patches, each flattened into a $d = 16 \times 16 \times 3 = 768$-dimensional continuous token, yielding 256 tokens per 256×256 image. Following prior work [Ke and Xue, 2026], we use 16 class-conditioning prefix tokens with dropout probability 0.1. Unless otherwise specified, we set the intermediate-state dimension to $d_z = 16$, the noise lower bound to $t_{\min} = 0.5$, and the token masking probability to $p_{\text{mask}} = 0.5$.

All main models are trained end-to-end on ImageNet-1K for 400 epochs. Unless otherwise specified, ablation experiments use the PRA-B scale and are trained for 100 epochs. The training objective combines a rectified-flow loss for AR generation in the intermediate-state space and a reconstruction loss for the pixel decoder. The reconstruction loss consists of an ℓ_1 pixel loss and an LPIPS loss on reconstructed images [Zhang et al., 2018]. We use equal weights for all loss terms without tuning. Optimization uses AdamW [Kingma, 2014, Loshchilov and Hutter, 2017] with batch size 512, $\beta = (0.9, 0.95)$, weight decay 0.05, a cosine learning-rate schedule with 20K warmup steps, peak learning rate 3×10^{-4}, and exponential moving average decay 0.9999. During inference, we generate each intermediate state with the diffusion head using a 100-step Euler-Maruyama solver. We use the linear classifier-free guidance (CFG) schedule from MAR and enable KV caching for efficient autoregressive generation.

6.2 Overall Image Generation Performance

Following prior work [Li and He, 2025], we generate 50K samples and report Fréchet Inception Distance (FID) [Heusel et al., 2017] as the primary fidelity metric, together with Inception Score (IS) [Salimans et al., 2016]. All metrics are computed using the ADM evaluation suite [Dhariwal and Nichol, 2021] against the standard ImageNet-256 reference statistics. For each model, we tune the classifier-free guidance (CFG) scale with a step size of 0.1 and report the best result.

We compare PRA with four families of methods along two axes: pixel-space versus two-stage generation, depending on whether a pretrained tokenizer or autoencoder is required, and autoregressive versus diffusion generation, depending on the generation paradigm. PRA falls into the pixel-space autoregressive family, since it directly models raw pixel patches as continuous tokens without relying on an external tokenizer.

Table 2 summarizes the results. PRA achieves a substantially better trade-off between generation quality and model scale than prior pixel-space AR methods. PRA-S, with only 135M parameters, reaches an FID of 2.58, outperforming much larger pixel-space AR models such as FARMER-1.9B/8, which has 1.9B parameters and an FID of 3.60. PRA also improves consistently with scale: PRA-B reaches an FID of 2.21, and PRA-L further improves it to 1.94. This establishes a new state of the art among pixel-space AR models, reducing the best reported FID in this family from 3.60 to 1.94. Compared with two-stage AR methods and pixel-space diffusion models, PRA remains competitive while directly modeling raw pixel tokens. These results show that PRA substantially strengthens pixel-space continuous-token AR, making it competitive with strong two-stage and diffusion-based image generators.

6.3 Output-Side Ablations: Learning AR-aligned Intermediate Targets

We first study the output-side component of PRA: replacing direct prediction of high-dimensional pixel patches with learned intermediate states. The goal is not to learn a generic image latent space, but to construct an intermediate target space that is easier for the current causal AR model to predict while preserving enough information to decode back to pixel patches. Table 3 ablates different aspects of this design: how the target space is constructed, and how large the intermediate states should be.

Table 2: Comparison of image generation performance on class-conditional ImageNet-1K at 256×256 resolution, reporting FID and IS.

Model	Params	FID↓	IS↑
Two-Stage Diffusion			
DiT-XL/2 [Peebles and Xie, 2023]	675+49M	2.27	278.2
SiT-XL/2 [Ma et al., 2024]	675+49M	2.06	277.5
REPA-XL/2 [Yu et al., 2024]	675+49M	1.42	305.7
LightningDiT-XL/2 [Yao et al., 2025]	675+49M	1.35	295.3
DDT-XL/2 [Wang et al., 2025b]	675+49M	1.26	310.6
RAE-XL/2 [Zheng et al., 2025a]	839+415M	**1.13**	262.6
Two-Stage Autoregressive			
VAR-d20 [Tian et al., 2024]	600M+40M	2.57	302.6
LlamaGen-XL [Sun et al., 2024a]	1.4B+70M	2.34	253.9
MAR-B [Li et al., 2024]	208+49M	2.31	281.7
MAR-L [Li et al., 2024]	479+49M	1.78	296.0
SphereAR-L [Ke and Xue, 2026]	479+49M	**1.54**	295.9
Pixel-Space Diffusion			
ADM-G [Dhariwal and Nichol, 2021]	554M	4.59	186.7
SiD [Hoogeboom et al., 2023]	2B	2.44	256.3
SiD2 [Hoogeboom et al., 2025]	N/A	**1.38**	–
PixelFlow-XL/4 [Chen et al., 2025]	677M	1.98	282.1
PixNerd-L/16 [Wang et al., 2025a]	458M	2.64	297.0
JiT-B/16 [Li and He, 2025]	131M	3.66	275.1
JiT-L/16 [Li and He, 2025]	459M	2.36	298.5
Pixel-Space Autoregressive			
JetFormer [Tschannen et al., 2025]	2.8B	6.64	–
FractalMAR-B [Li et al., 2025]	186M	11.8	274.3
FractalMAR-L [Li et al., 2025]	438M	7.30	334.9
FractalMAR-H [Li et al., 2025]	848M	6.15	348.9
FARMER-1.1B/8 [Zheng et al., 2025b]	1.1B	5.02	237.0
FARMER-1.9B/8 [Zheng et al., 2025b]	1.9B	3.60	269.2
PRA-S (ours)	135M	2.58	273.9
PRA-B (ours)	250M	2.21	276.9
PRA-L (ours)	511M	**1.94**	287.3

Table 3: Ablations on learned intermediate targets.

Factor	Variant	FID↓	IS↑
Target	LDM enc.	3.37	266.73
	Local only	3.08	271.96
	Prefix aware	**2.88**	279.05
Masking	$p_{\text{mask}} = 0.0$	2.96	280.32
	$p_{\text{mask}} = 0.3$	**2.79**	280.66
	$p_{\text{mask}} = 0.5$	2.88	279.05
	$p_{\text{mask}} = 0.7$	3.01	279.03
Dim.	$d_z = 8$	3.36	285.85
	$d_z = 16$	**2.88**	279.05
	$d_z = 32$	3.50	297.17
	$d_z = 64$	7.03	220.11

Table 4: Ablations on AR training inputs.

Input	FID ↓	IS ↑
GT pixel	42.36	72.50
GT pixel + noise	32.60	110.60
GT Inter. states	3.21	261.86
GT Inter. states + noise	3.05	268.34
Decoded Pixel (PRA)	**2.88**	279.05

Table 5: Ablations on robust pixel inputs.

Noise level	FID ↓	IS ↑
$t_{\min} = 1.0$	3.35	283.60
$t_{\min} = 0.7$	**2.62**	282.03
$t_{\min} = 0.5$	2.88	279.05
$t_{\min} = 0.3$	2.96	291.34
$t_{\min} = 0.0$	3.06	291.01

Table 6: ImageNet linear probing results. PRA-L achieves highest top-1 accuracy.

Paradigm	Space	Model	Param.	Top-1 Acc. (%) ↑
Diffusion	Latent	DiT-XL/2	675+49M	43.28
	Pixel	JiT-L	459M	42.76
AR	Latent	SphereAR-L	479+49M	52.19
	Pixel	**PRA-L**	511M	**68.80**

For target construction, as shown in Table 3, a frozen LDM encoder [Rombach et al., 2022] performs worse than the learned PRA target, indicating that an off-the-shelf latent space is not necessarily suitable for causal AR prediction. A local-only target, which encodes only the current pixel patch, also underperforms the prefix-aware target. The prefix-aware target incorporates the causal prefix representation $\mathbf{z}_i = g_\psi(\mathbf{x}_i, \mathbf{h}_{i-1})$ and performs best, improving FID from 3.08 to 2.88, showing that the intermediate target should be adapted to the causal context, rather than serving merely as a local autoencoding code. Encoder masking further regularizes this target construction. Without masking, the encoder can rely too heavily on the clean current patch, which weakens the dependence of the target on the causal prefix. Moderate token masking encourages the intermediate state to use prefix information and improves its compatibility with AR prediction, while overly strong masking removes too much local information and makes decoding harder. We therefore use the prefix-aware target with $p_{\text{mask}} = 0.5$ by default.

We next vary the intermediate-state dimension d_z, which controls the trade-off between predictability and information preservation. If the state is too small, such as $d_z = 8$, it lacks sufficient capacity to reconstruct pixel patches. If the state is too large, the AR model again faces a harder continuous-token prediction problem. This is reflected by the degradation at $d_z = 32$ and especially $d_z = 64$. The best performance is obtained at $d_z = 16$, suggesting that a compact but expressive intermediate space is crucial for reducing single-step errors while maintaining pixel decodability.

6.4 Input-Side Ablations: Inference-Like Pixel Inputs

We next ablate the AR training inputs, which target the input-side train–inference mismatch. As shown in Table 4, using clean ground-truth pixels as inputs performs poorly, even though the output target has already been moved to the low-dimensional intermediate space. This is because inference does not condition on ground-truth pixels, but on pixel tokens decoded from generated intermediate states. These decoded pixels can deviate substantially from real pixels, creating a strong input mismatch. Pixel-space noise injection improves robustness, but independent perturbations still do not match the structured artifacts introduced by the intermediate-state-to-pixel path.

We then consider using ground-truth(GT) intermediate states as AR inputs. Without noise, this variant still suffers from a mismatch, since training conditions on target states constructed from ground-truth tokens, whereas inference must rely on states produced by the model. Adding noise to the GT intermediate states makes the inputs closer to generated states and substantially improves performance, indicating that the intermediate space is easier for autoregressive modeling. However, this variant changes the external AR interface from pixel inputs to learned-state inputs, making the model closer to latent-space AR. In contrast, Decoded Pixel (PRA) preserves the pixel-in, pixel-out interface while better matching the inference condition. It constructs training inputs by perturbing intermediate states and mapping them through the same pixel decoder used at inference. This provides a parallel approximation to autoregressive rollout inputs: each position is constructed independently, but through the same intermediate-state-to-pixel path used during sequential inference. As a result, the AR model is trained on decoded pixel inputs that are closer to inference-time tokens than clean pixels, independently noised pixels, or GT intermediate states.

We further study the perturbation strength used to construct decoded pixel inputs in Table 5. The lower bound $t_{\min}$ controls how close the perturbed state is to the target intermediate state: larger values produce cleaner inputs, while smaller values introduce stronger perturbations. Both extremes can be suboptimal. If the inputs are too clean, training remains close to teacher forcing; if they are too corrupted, the causal context becomes unreliable. A moderate value provides the best trade-off between inference-likeness and prefix reliability. We use $t_{\min} = 0.5$ as the default unless otherwise specified.

6.5 Image Understanding via Linear Probing

A potential advantage of pixel-space autoregressive modeling is that the model operates directly on raw visual tokens, which may preserve useful information for visual understanding. We evaluate this by applying the standard ImageNet-1K linear probing protocol [Chen et al., 2020] to the generation-trained backbone. The pretrained model is frozen, and only a linear classifier is trained.

We evaluate PRA-L directly, without architectural modifications or additional unsupervised pretraining. As shown in Table 6, PRA-L achieves a top-1 accuracy of 68.80%, substantially outperforming both the latent-space AR baseline SphereAR-L and the pixel-space diffusion baseline JiT-L. This suggests that PRA not only improves pixel-space generation, but also learns more transferable visual representations, supporting its potential as a unified pixel-space model for image generation and understanding.

7 Conclusion

We presented *Parallel Rollout Approximation* (PRA) for pixel-space autoregressive image generation. PRA addresses two coupled challenges of high-dimensional continuous-token AR: large single-step errors from directly generating pixel patches, and cross-step error accumulation caused by train–inference mismatch. To reduce output-side generation difficulty, PRA learns low-dimensional intermediate targets end-to-end and maps them back to pixel-space tokens with a pixel decoder. To reduce the input-side mismatch, PRA constructs inference-like pixel inputs in parallel through the same decode-to-pixel path used at inference, avoiding costly sequential rollouts while preserving a pixel-in, pixel-out AR interface. Experiments on class-conditional ImageNet-1K generation show that

PRA substantially improves pixel-space AR generation and establishes a new state of the art among pixel-space AR models. Beyond generation, linear probing results suggest that PRA also learns transferable visual representations, supporting its potential for unified pixel-space image generation and understanding.

Limitations. PRA introduces an internal intermediate state, a pixel decoder, and an additional parallel AR forward during training. Although these components are trained end-to-end without a separately pretrained tokenizer, simplifying the framework and validating it across broader data domains and generation tasks remain important directions for future work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=3zKtaqxLhW`.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PmLR, 2020.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35: 16344–16359, 2022.

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

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

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

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, 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.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pages 13213–13232. PMLR, 2023.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion: 1.5 fid on imagenet512 with pixel-space diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18062–18071, 2025.

Guolin Ke and Hui Xue. Hyperspherical latents improve continuous-token autoregressive generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=H13wHRiL3i.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

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.

Tianhong Li, Qinyi Sun, Lijie Fan, and Kaiming He. Fractal generative models. *arXiv preprint arXiv:2502.17437*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Shuqi Lu, Haowei Lin, Lin Yao, Zhifeng Gao, Xiaohong Ji, Linfeng Zhang, Guolin Ke, et al. Uni-3dar: Unified 3d generation and understanding via autoregression on compressed spatial tokens. *arXiv preprint arXiv:2503.16278*, 2025.

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

Lingwei Meng, Long Zhou, Shujie Liu, Sanyuan Chen, Bing Han, Shujie Hu, Yanqing Liu, Jinyu Li, Sheng Zhao, Xixin Wu, et al. Autoregressive speech synthesis without vector quantization. *arXiv preprint arXiv:2407.08551*, 2024.

Marco Pasini, Javier Nistal, Stefan Lattner, and George Fazekas. Continuous autoregressive models with noise augmentation avoid error accumulation, 2024. URL https://arxiv.org/abs/2411.18447.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, October 2023.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

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

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Florian Schmidt. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 157–167, 2019.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024a.

Yutao Sun, Hangbo Bao, Wenhui Wang, Zhiliang Peng, Li Dong, Shaohan Huang, Jianyong Wang, and Furu Wei. Multimodal latent language modeling with next-token diffusion. *arXiv preprint arXiv:2412.08635*, 2024b.

NextStep Team, Chunrui Han, Guopeng Li, Jingwei Wu, Quan Sun, Yan Cai, Yuang Peng, Zheng Ge, Deyu Zhou, Haomiao Tang, et al. Nextstep-1: Toward autoregressive image generation with continuous tokens at scale. *arXiv preprint arXiv:2508.10711*, 2025.

Hansi Teng, Hongyu Jia, Lei Sun, Lingzhi Li, Maolin Li, Mingqiu Tang, Shuai Han, Tianning Zhang, WQ Zhang, Weifeng Luo, et al. Magi-1: Autoregressive video generation at scale. *arXiv preprint arXiv:2505.13211*, 2025.

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.

Michael Tschannen, Cian Eastwood, and Fabian Mentzer. Givt: Generative infinite-vocabulary transformers. In *European Conference on Computer Vision*, pages 292–309. Springer, 2024.

Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. Jetformer: An autoregressive generative model of raw images and text, 2025. URL `https://arxiv.org/abs/2411.19722`.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. *arXiv preprint arXiv:2507.23268*, 2025a.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. *arXiv preprint arXiv:2504.05741*, 2025b.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15703–15712, 2025.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025a.

Guangting Zheng, Qinyu Zhao, Tao Yang, Fei Xiao, Zhijie Lin, Jie Wu, Jiajun Deng, Yanyong Zhang, and Rui Zhu. Farmer: Flow autoregressive transformer over pixels. *arXiv preprint arXiv:2510.23588*, 2025b.

A Model Details

Table 7: PRA scale configurations. "res / AdaLN" denotes the number of diffusion-head residual blocks and the number of shared AdaLN groups. The diffusion head width matches d. The pixel-decoder width d_r also defines the encoder g_ψ width. All blocks use an MLP ratio of 4 with SwiGLU. Training cost is reported as wall-clock days on 8×A100 GPUs for 400-epoch training.

Model	AR Transformer f_θ			Diffusion head v_ω		Pixel decoder D_ϕ			Params.	Training Cost by
	layers	heads	d	res / AdaLN	width	layers	heads	d_r		8×A100 (days)
PRA-S	12	12	768	4 / 1	768	6	8	512	135M	3.125
PRA-B	24	12	768	6 / 2	768	6	12	768	250M	6
PRA-L	30	16	1024	8 / 2	1024	8	12	768	511M	14.3

A.1 Architecture Overview

PRA consists of four modules, all trained end-to-end: a causal AR Transformer f_θ, an intermediate-state encoder g_ψ that produces $\mathbf{z}_i \in \mathbb{R}^{d_z}$, a pixel decoder D_ϕ that maps a noisy intermediate state back to a continuous pixel patch, and a diffusion head v_ω that models the rectified-flow distribution of $\mathbf{z}_i$ conditioned on the AR hidden state. The input image is patchified into non-overlapping 16×16 patches in raster-scan order. For a 256×256 RGB image, this yields $T = 256$ tokens, each corresponding to a $16\times16\times3 = 768$-dimensional pixel patch. The scale-specific architectural configurations and training costs are summarized in Table 7.

A.2 Causal AR Transformer f_θ

The backbone is a causal Transformer [Vaswani et al., 2017] with pre-RMSNorm attention [Zhang and Sennrich, 2019], SwiGLU feed-forward layers [Shazeer, 2020], and 2-D rotary position embeddings (RoPE) [Su et al., 2024] over patch coordinates. Class conditioning is injected through $K{=}16$ learnable prefix tokens, obtained from an embedding table indexed by the class label and prepended to the patch sequence. We use FlashAttention [Dao et al., 2022] during training. Class dropout with probability 0.1 replaces the class label with a null label, enabling classifier-free guidance at inference.

A.3 Intermediate-State Encoder g_ψ

The intermediate-state encoder maps each pair $(\mathbf{x}_i, \mathbf{h}_{i-1})$ to a low-dimensional intermediate state $\mathbf{z}_i$. We use $d_z{=}16$ for all scales. The encoder first linearly projects the pixel patch to the encoder width d_r, then applies four SwiGLU residual blocks with LayerNorm and AdaLN modulation conditioned on $\mathbf{h}_{i-1}$. The conditioning representation is the AR hidden state after RMSNorm and an additive learnable position embedding. AdaLN parameters are shared every two residual blocks. The encoder hidden width matches the pixel-decoder width d_r in Table 7.

To encourage $\mathbf{z}_i$ to depend on causal context rather than only the current patch, we use two-level encoder masking during training. With probability p_{sample}, a training example is selected for masking; within selected examples, each token is replaced by a learned mask embedding with probability p_{token}. Unless otherwise specified, we use $p_{\text{sample}}{=}0.9$ and $p_{\text{token}}{=}0.5$.

A.4 Pixel Decoder D_ϕ

The pixel decoder is a causal Transformer that maps noisy intermediate states back to pixel patches. It consists of three components: (i) a linear projection from $\mathbb{R}^{d_z}$ to $\mathbb{R}^{d_r}$, (ii) a causal Transformer with the same block design as the AR backbone but separate weights, with depth and width specified in Table 7, and (iii) an RMSNorm followed by a two-layer SiLU MLP head that projects the hidden representation back to a $16\times16\times3 = 768$-dimensional pixel patch. Causality is preserved so that KV caching can be used; at inference, D_ϕ is invoked once per autoregressive step. The final output linear layer is zero-initialized.

A.5 Diffusion Head v_ω

The diffusion head models the rectified-flow distribution of $\mathbf{z}_i$ conditioned on the AR hidden state. It is implemented as a SwiGLU MLP stack with shared AdaLN modulation. The conditioning vector is $\mathbf{c}_i = \mathrm{RMSNorm}(\mathbf{h}_{i-1}) + \mathbf{p}_i^{\mathrm{diff}}$, where $\mathbf{p}_i^{\mathrm{diff}}$ is a learned position embedding. For flow timestep s, the modulation input is $\mathrm{SiLU}(t_{\text{embed}}(s) + \mathbf{c}_i)$, where t_{embed} is a sinusoidal timestep embedding. The head predicts the rectified-flow velocity. Both the final output linear layer and the AdaLN modulation linear layer are zero-initialized.

B Training Objective and Implementation Details

PRA uses parallel training while exposing the AR model to inputs that better match inference-time generated tokens. The implementation performs a reconstruction pass and an AR training pass. In the reconstruction pass, ground-truth pixel patches are encoded into intermediate states, optionally perturbed in intermediate space, decoded back to pixel patches, and supervised by image reconstruction losses. In the AR pass, the decoded pixel patches are stop-gradient inputs to the causal AR Transformer, and the diffusion head is trained to generate the clean intermediate targets.

Intermediate-space perturbation. During training, the decoder input is a perturbed version of the target intermediate state. For selected samples, we draw $\epsilon_i \sim \mathcal{N}(0, I)$ and $t_i \sim \mathcal{U}(t_{\min}, 1)$ and form

$$\tilde{\mathbf{z}}_i = t_i \mathbf{z}_i + (1 - t_i)\epsilon_i. \tag{4}$$

Unless otherwise specified, we perturb a sample with probability 0.9 and use $t_{\min}$=0.5 for the first 350 epochs, then switch to $t_{\min}$=0.7 for the remaining 50 epochs. The same perturbed latent sequence is used by the pixel decoder to produce the decoded pixel inputs for the AR pass. We apply LayerNorm without affine parameters to intermediate states before diffusion-head supervision and after diffusion-head sampling.

Clean targets under encoder masking. When encoder masking is active, the implementation uses a clean no-mask encoder pass to define the detached target $\mathbf{z}_i$ for the AR diffusion loss, while the masked/noised path is used to train the decoder and construct decoded pixel inputs. The AR loss therefore does not backpropagate through the detached target-producing encoder path; the encoder and decoder are trained through reconstruction, perceptual, and auxiliary representation objectives.

Encoder-side gradient scaling. The reconstruction and autoregressive generation objectives share the same AR backbone. To balance the training signal strength, we scale the gradients flowing from the encoder-side reconstruction and auxiliary losses into the shared AR backbone by a constant factor. Unless otherwise specified, we use a gradient scale of 0.3.

AR rectified-flow loss. For each token target $\mathbf{z}_i$, we draw $\epsilon_i \sim \mathcal{N}(0, I)$ and $s_i \sim \sigma(\mathcal{N}(\mu, 1))$, where σ is the sigmoid function and μ is a timestep-shift hyperparameter. We train the diffusion head on the linear interpolation $\mathbf{z}_i^{s_i} = (1 - s_i)\epsilon_i + s_i \mathbf{z}_i$:

$$\mathcal{L}_{\mathrm{AR}} = \frac{1}{T} \sum_{i=1}^{T} \left\| v_\omega(\mathbf{z}_i^{s_i}, s_i, \bar{\mathbf{h}}_{i-1}) - (\mathbf{z}_i - \epsilon_i) \right\|_2^2, \tag{5}$$

where $\bar{\mathbf{h}}_{i-1}$ is computed from the decoded prefix $\bar{\mathbf{x}}_{<i}$ rather than the clean prefix. We use μ=0 in the main experiments.

Reconstruction and auxiliary losses. The pixel decoder is supervised on the reconstructed image with an ℓ_1 reconstruction loss and an LPIPS perceptual loss [Zhang et al., 2018]. We also use an auxiliary representation loss that predicts per-patch normalized pixel values from the AR hidden states produced in the reconstruction pass. The overall objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{AR}} + \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{lpips}}\mathcal{L}_{\mathrm{lpips}} + \lambda_{\mathrm{repr}}\mathcal{L}_{\mathrm{repr}}. \tag{6}$$

For the main runs, we use λ_{rec}=1, λ_{lpips}=1, and λ_{repr}=1 unless otherwise stated in ablations.

Ground-truth pixel replacement. We replace a small fraction of decoded AR inputs with noised ground-truth pixel patches. This stabilizes training while preserving the decoded-input training distribution. We set the replacement probability is 0.04 by default.

C Sampling Details

At inference time, generation proceeds sequentially in raster order. At step i, the AR Transformer consumes the previously generated pixel patch $\hat{\mathbf{x}}_{i-1}$, produces a hidden state, the diffusion head samples $\hat{\mathbf{z}}_i$, and the pixel decoder maps $\hat{\mathbf{z}}_i$ to $\hat{\mathbf{x}}_i$. We use KV caching for both the AR Transformer and the causal pixel decoder.

For the velocity-parameterized diffusion head used in the main experiments, sampling starts from Gaussian noise at $s = 0$ and integrates to $s = 1$ with 100 Euler–Maruyama steps followed by a final deterministic Euler step. We use classifier-free guidance by evaluating conditional and null-label predictions in the diffusion head, and apply a linear guidance schedule over autoregressive positions. As in training, the generated intermediate state is normalized before being passed to the pixel decoder.

D Training Hyperparameters

All main ImageNet-1K models are trained for 400 epochs with global batch size 512. We use AdamW with peak learning rate 3×10^{-4}, β_1=0.9, β_2=0.95, weight decay 0.05, gradient clipping at norm 1.0, a cosine learning-rate schedule, and 20K warmup steps. We maintain exponential moving averages (EMA) of model weights with decay 0.9999. Unless otherwise specified, class dropout is 0.1, the number of class prefix tokens is 16, the patch size is 16, and the intermediate dimension is d_z=16.