PIXELMOE: SCALING PIXEL-SPACE DIFFUSION WITH LLM-STYLE MIXTURE-OF-EXPERTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pixel-space diffusion models provide an appealing alternative to latent-space diffusion models by directly modeling raw images without relying on a learned tokenizer. However, scaling pixel-space models has predominantly followed dense strategies, tightly coupling model capacity with active computation. In this work, we show that standard LLM-style Mixture-of-Experts (MoE) architectures can be effectively transferred to pixel-space diffusion. Building on this finding, we introduce **PixelMoE**, a pixel-space generative framework that sparsely scales model capacity with standard LLM-style MoE, without diffusion-specific routing mechanisms or auxiliary routing objectives. Specifically, PixelMoE combines standard Token-Choice (TC) routing with fine-grained routed experts for input-adaptive sparse computation, a shared expert for capturing common representations, and a bias-based load-balancing strategy. To complement sparse capacity scaling, PixelMoE further introduces Pixel-Space Hierarchical Flow Matching, which varies spatial resolution along the denoising trajectory from global structure formation to fine-grained detail refinement. At comparable active parameters, PixelMoE consistently outperforms its dense counterparts across model scales, demonstrating effective sparse capacity scaling in pixel space. On class-conditional ImageNet-1K, PixelMoE achieves FID scores of 1.49 at 256×256 and 1.56 at 512×512 without representation alignment or perceptual losses, comparing favorably with leading pixel-space and latent-space generative models.

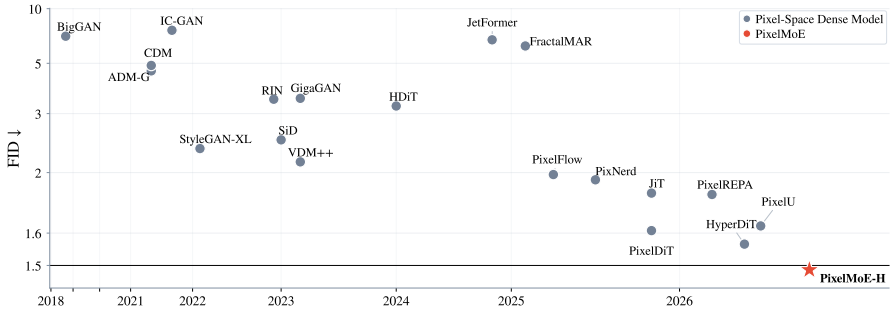


Figure 1: **Progress in pixel-space image generation on ImageNet-1K at** 256×256**.** PixelMoE sparsely scales model capacity with MoE and compares favorably with leading approaches.

1 INTRODUCTION

Recent advances in diffusion models, particularly Diffusion Transformers (DiTs) (Peebles & Xie, 2023; Ma et al., 2024) trained with flow matching objectives (Lipman et al., 2023; Liu et al., 2023), have substantially improved the quality and scalability of image generation. Most modern diffusion models (Rombach et al., 2022) denoise in latent spaces extracted by variational autoencoders (VAEs) (Kingma & Welling, 2013), whose lossy compression may discard high-frequency details and introduce structural artifacts, leading to a reconstruction bottleneck that limits achievable image fidelity. Recently, pixel-space diffusion models (Dhariwal & Nichol, 2021; Li & He, 2026) have regained attention by directly modeling raw images rather than learned latent representations. Advances in clean-image prediction (Li & He, 2026), architectural design (He et al., 2026; Yu et al., 2026), and training objectives (Ma et al., 2026b) have made raw-pixel modeling easier, enabling pixel-space models to achieve performance competitive with leading latent-space approaches.

Despite this progress, sparsely scaling pixel-space diffusion remains largely unexplored. Existing pixel-space models primarily scale their generators densely by increasing model width and depth, tightly coupling capacity growth with active computation. In contrast, Mixture-of-Experts (MoE)

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

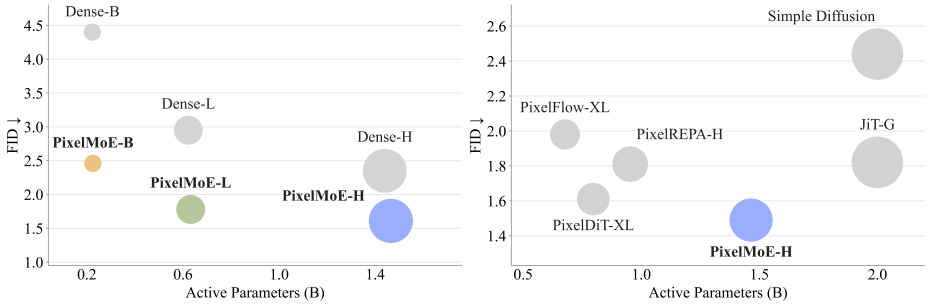


Figure 2: **ImageNet-1K generation with PixelMoE.** Left: Dense *vs.* PixelMoE after 100 training epochs, where PixelMoE consistently achieves lower FID at comparable active parameters. Right: Comparison with representative pixel-space methods. Marker size denotes active parameters.

architectures (Shazeer et al., 2017) expand total capacity through sparse expert activation while controlling active computation, and have become a standard scaling paradigm in Large Language Models (LLMs) (Liu et al., 2024). However, directly transferring standard LLM-style MoE to diffusion has not consistently yielded clear gains over dense counterparts (Fei et al., 2024; Liu et al., 2025). Subsequent approaches therefore adopt alternative routing paradigms such as Expert-Choice (EC) (Sun et al., 2025; Shi et al., 2025), or introduce diffusion-specific routing guidance and objectives (Wei et al., 2026). This naturally raises the question: ***Can standard LLM-style MoE be directly transferred to pixel-space diffusion?***

We answer this question affirmatively. We propose **PixelMoE**, a **Pixel-space Mixture-of-Experts** generative framework that sparsely scales pixel-space diffusion transformers without diffusion-specific routing mechanisms or auxiliary routing objectives. Specifically, PixelMoE adopts standard Token-Choice (TC) routing, where each image token dynamically selects its top-A experts based on the routing scores. Each MoE layer combines fine-grained routed experts for input-adaptive sparse computation with a shared expert that captures common representations across image tokens. To maintain balanced expert utilization without auxiliary balancing objectives, we further adopt a bias-based load-balancing strategy. Beyond sparse capacity scaling, pixel-space representations can be directly downsampled and upsampled across resolutions while preserving meaningful visual structure. We exploit this property with Pixel-Space Hierarchical Flow Matching, organizing generation into a low-resolution coarse stage followed by full-resolution refinement.

Extensive experiments demonstrate strong sparse scaling behavior of PixelMoE. Under comparable active parameters, PixelMoE outperforms its dense counterparts across model scales. Furthermore, increasing the number of routed experts consistently improves generation quality. Standard Token-Choice routing also substantially outperforms Expert-Choice routing, without diffusion-specific routing guidance. Moreover, our routing analysis reveals clear expert specialization across semantic classes and along the denoising trajectory without explicit semantic or timestep guidance. Together, these results suggest that effective sparse scaling does not require diffusion-specific routing: with appropriately structured sparse capacity, standard Token-Choice routing can effectively exploit the rich semantic and denoising information already encoded in pixel-space representations. On ImageNet-1K (Deng et al., 2009), PixelMoE achieves FID scores of 1.49 at 256×256 and 1.56 at 512×512 without representation alignment (Yu et al., 2025; Shin et al., 2026) or perceptual losses (Ma et al., 2026b), comparing favorably with leading pixel-space and latent-space generative models. These results demonstrate that LLM-style sparse scaling transfers effectively to pixel-space diffusion.

2 Related Work

Pixel-Space Generative Models. Early generative models, including GANs (Brock et al., 2019; Casanova et al., 2021) and diffusion models (Dhariwal & Nichol, 2021; Ho et al., 2022), synthesize images directly in pixel space. Latent-space models (Rombach et al., 2022; Esser et al., 2021) later became dominant by modeling compact representations extracted by variational autoencoders (VAEs) (Kingma & Welling, 2013), whose lossy compression introduces a reconstruction bottleneck that limits the achievable generation fidelity. This limitation has renewed interest in pixel-space generative models, which directly model raw images without lossy compression. Simple Diffusion (Hoogeboom et al., 2023) demonstrated competitive high-resolution pixel-space generation through architectural scaling, noise schedules, and spatial downsampling, while VDM++ (Kingma & Gao, 2023) studied diffusion-objective weighting and noise augmentation. JiT (Li & He, 2026)

further showed that a large-patch Transformer can effectively model raw pixels through clean-image prediction. Subsequent approaches improve pixel-space generation through multi-resolution generation (Chen et al., 2025), neural-field decoding (Wang et al., 2026a), hierarchical architectures (Yu et al., 2026; He et al., 2026; Guo et al., 2026), frequency-decoupled modeling (Ma et al., 2026a), representation alignment (Shin et al., 2026), perceptual supervision (Ma et al., 2026b), and alternative flow parameterizations (Chen et al., 2026a). Despite these advances, most existing pixel-space models remain dense, coupling increased model capacity with increased active computation, while sparse MoE scaling in pixel-space diffusion remains underexplored.

Mixture-of-Experts in Visual Generation. The Mixture-of-Experts (MoE) paradigm was introduced to enable specialization among multiple expert networks (Jacobs et al., 1991) and later extended with sparse gating (Shazeer et al., 2017). Modern MoE architectures (Fedus et al., 2022; Dai et al., 2024) have become a key scaling mechanism for Large Language Models (LLMs) (Yang et al., 2025; Liu et al., 2024), motivating their adoption in visual generation. Early expert-based diffusion models, largely built on U-Net backbones, partition the denoising trajectory by noise level, denoising task, or generation path and assign specialized parameters to different regimes (Balaji et al., 2022; Feng et al., 2023; Xue et al., 2023; Lee et al., 2024; Ganjdanesh et al., 2024). As diffusion backbones shift toward Diffusion Transformers (DiTs) (Peebles & Xie, 2023), recent work instead explores sparse expert routing within Transformer blocks. However, directly applying conventional Token-Choice routing has not consistently yielded clear gains over dense models (Fei et al., 2024). Subsequent approaches have therefore introduced Expert-Choice routing (Sun et al., 2025), diffusion-aware routing or dynamic capacity allocation (Cheng et al., 2025; Shi et al., 2025), and specialized token-expert assignment with auxiliary routing objectives (Yuan et al., 2025; Wei et al., 2026). However, most existing sparse DiT approaches focus on latent-space diffusion. JiT-MoE (Liu et al., 2025) has recently extended MoE to pixel-space diffusion but reports only modest gains over its dense JiT counterparts. We therefore investigate whether standard LLM-style sparse MoE can provide systematic scaling gains in pixel-space generation.

3 PIXELMOE: PIXEL-SPACE DIFFUSION WITH MIXTURE-OF-EXPERTS

PixelMoE is a pixel-space generative framework designed to sparsely scale model capacity with standard LLM-style Mixture-of-Experts (MoE), while employing Pixel-Space Hierarchical Flow Matching for coarse-to-fine generation. As illustrated in the left panel of Figure 3, generation proceeds through two stages at different spatial resolutions. Stage 0 performs partial denoising at low resolution to establish global structure, while Stage 1 continues denoising at full resolution for detail refinement. The two stages share the PixelMoE backbone and Decoupled Diffusion Transformer (DDT) (Wang et al., 2026b) prediction head. The backbone consists of initial dense blocks followed by sparse PixelMoE blocks, while the DDT head performs stage-wise endpoint prediction.

As shown in the right panel of Figure 3, each PixelMoE block consists of a self-attention layer followed by an LLM-style sparse MoE layer. For each image token, standard Token-Choice routing selects a subset of fine-grained routed experts, while a shared expert is always activated to capture common representations across image tokens. To maintain balanced expert utilization without an auxiliary load-balancing objective, we further introduce a balancing bias during routing. Importantly, the router operates directly on pixel-token representations without explicit semantic or timestep routing guidance. We detail the sparse MoE design and the hierarchical flow formulation in the following subsections.

3.1 LLM-STYLE MIXTURE-OF-EXPERTS FOR PIXEL-SPACE DIFFUSION

The core design of PixelMoE is to directly transfer the standard LLM-style MoE paradigm to pixel-space diffusion. Following modern LLM-style MoE architectures (Liu et al., 2024; Team et al., 2026), we retain dense feed-forward networks (FFNs) in the early Transformer blocks and replace the FFNs in the remaining blocks with sparse MoE layers. Each MoE layer adopts standard Token-Choice routing over pixel-token representations, combines fine-grained routed experts with a shared expert, and employs a bias-based load-balancing strategy without an auxiliary objective.

Token-Choice Routing. For each image token, Token-Choice (TC) routing first computes its affinity to all routed experts. Let $\mathbf{h}_{t,i}^{s} \in \mathbb{R}^{d}$ denote the hidden state of image token i at global timestep t in stage s, and let E denote the number of routed experts. We use a bias-free linear router with

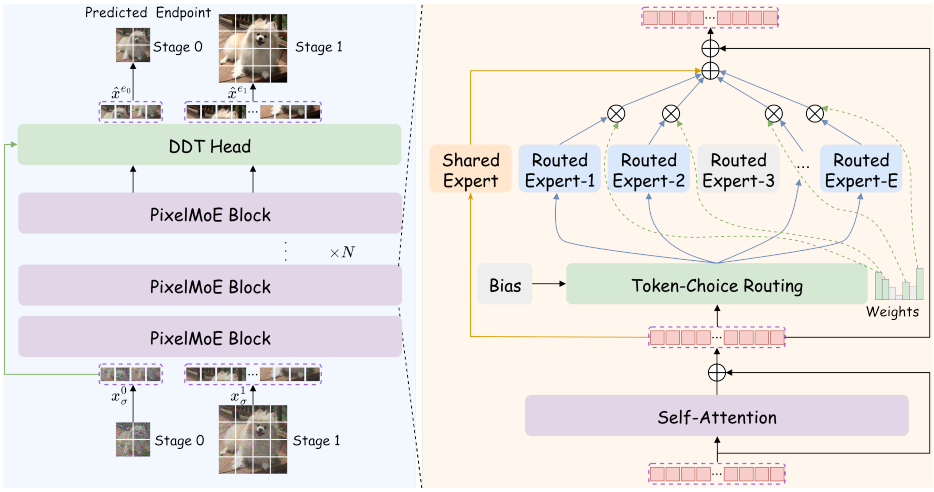


Figure 3: **Overview of PixelMoE.** Left: PixelMoE performs Pixel-Space Hierarchical Flow Matching using a shared PixelMoE backbone and DDT prediction head; the initial dense blocks are omitted for clarity. Right: Each PixelMoE block consists of self-attention followed by an LLM-style sparse MoE layer with Token-Choice routed experts and a shared expert. A balancing bias is employed to maintain balanced expert utilization. RMSNorm and timestep/class conditioning are omitted.

weight $\mathbf{W}_r \in \mathbb{R}^{E\times d}$ to produce sigmoid routing scores:

$$\mathbf{r}^s_{t,i} = \mathrm{sigmoid}\left(\mathbf{W}_r \mathbf{h}^s_{t,i}\right), \tag{1}$$

where $r^s_{t,i,e}$ denotes the routing score of token i for the e-th routed expert. To regulate expert utilization, each routed expert is associated with a balancing bias b_e, which is added to the routing score only during expert selection. The set of top-A activated experts is

$$\mathcal{S}^s_{t,i} = \mathrm{TopA}_e\left(r^s_{t,i,e} + b_e\right), \tag{2}$$

where A denotes the number of routed experts activated for each image token. After selection, the balancing bias is no longer used and the routing scores are normalized over the selected experts:

$$g^s_{t,i,e} = \frac{r^s_{t,i,e}}{\sum_{j\in\mathcal{S}^s_{t,i}} r^s_{t,i,j}}, \qquad e \in \mathcal{S}^s_{t,i}, \tag{3}$$

where $g^s_{t,i,e}$ denotes the routing weight of expert e for token i. Notably, the router relies solely on image-token hidden states, without explicit semantic or timestep guidance.

Expert Aggregation. Given the selected experts $\mathcal{S}^s_{t,i}$ and routing weights $g^s_{t,i,e}$, PixelMoE aggregates the routed-expert outputs with the output of an always-active shared expert. Let $\mathcal{E}_e(\cdot)$ denote the e-th routed expert and $\mathcal{E}_{\mathrm{sh}}(\cdot)$ the shared expert. Only the experts in $\mathcal{S}^s_{t,i}$ process the image-token hidden state $\mathbf{h}^s_{t,i}$ in the routed branch, while the shared expert is applied to every image token. Their outputs are combined as

$$\mathbf{y}^s_{t,i} = \lambda_r \sum_{e\in\mathcal{S}^s_{t,i}} g^s_{t,i,e}\, \mathcal{E}_e\left(\mathbf{h}^s_{t,i}\right) + \mathcal{E}_{\mathrm{sh}}\left(\mathbf{h}^s_{t,i}\right), \tag{4}$$

where $\lambda_r = \sqrt{A}$ denotes the routing scale. Both routed and shared experts are implemented as FFNs with reduced intermediate dimensions, such that their combined active parameter count is comparable to that of the dense baseline.

Bias-Based Load Balancing. To maintain balanced expert utilization without introducing an auxiliary load-balancing objective, we dynamically update the expert-wise balancing biases during training. We collect the balancing biases for the routed experts into $\mathbf{b} = (b_1, \ldots, b_E) \in \mathbb{R}^E$. The expert-load distribution in the current training batch is $\mathbf{F} \in \mathbb{R}^E$, obtained by averaging image-token assignments, with each selected expert contributing $1/A$, while the uniform target is $\mathbf{Q} = (1/E, \ldots, 1/E)$. We update the balancing biases as

$$\mathbf{b} \leftarrow \mathbf{b} - \gamma \frac{\mathbf{F} - \mathbf{Q}}{\mathrm{RMS}(\mathbf{F} - \mathbf{Q})}, \tag{5}$$

where $\gamma = 0.001$ controls the update magnitude. The detailed computation of $\mathbf{F}$ and the RMS normalization is provided in Section A.

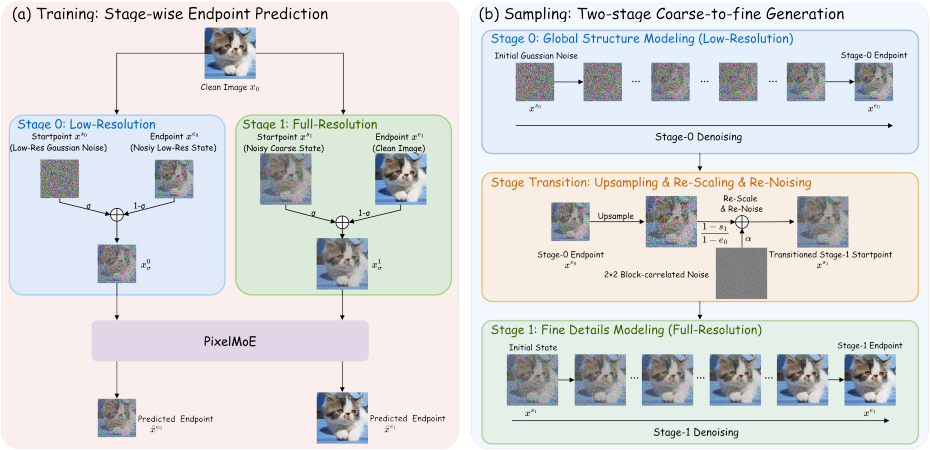


Figure 4: **Pixel-Space Hierarchical Flow Matching in PixelMoE.** PixelMoE is jointly trained on a low-resolution coarse stage and a full-resolution refinement stage with stage-wise endpoint prediction. During sampling, generation first follows the low-resolution Stage 0 flow to obtain a coarse endpoint, which is then transitioned to the Stage 1 starting state for full-resolution refinement.

3.2 Pixel-Space Hierarchical Flow Matching

Pixel space naturally supports coarse-to-fine generation, as spatial resampling directly produces meaningful multi-resolution images. We therefore vary the spatial resolution along the denoising trajectory, using coarse representations during early denoising and full-resolution representations for later refinement. This hierarchical trajectory also requires the model to process heterogeneous pixel-token representations across resolutions and noise levels, which are naturally accommodated by the input-adaptive routing of PixelMoE. Specifically, Pixel-Space Hierarchical Flow Matching organizes generation into two stages: Stage 0 performs partial denoising at low resolution for coarse structure modeling, while Stage 1 continues at full resolution for detail refinement. As illustrated in Figure 4, both stages share the same PixelMoE model while denoising at different spatial resolutions and noise levels.

Hierarchical Flow Trajectory. Given a clean image x_0, Stage 0 performs partial denoising at low resolution, starting from Gaussian noise x^{s_0} and terminating at a noisy coarse endpoint x^{e_0}:

$$\begin{aligned} x^{s_0} &= \epsilon_0, \qquad \epsilon_0 \sim \mathcal{N}(0, \delta_n^2 I), \\ x^{e_0} &= (1 - t_{e_0})\mathcal{D}(x_0) + t_{e_0}\epsilon_0, \end{aligned} \tag{6}$$

where $\mathcal{D}(\cdot)$ denotes $2\times$ bilinear downsampling and t_{e_0} specifies the Stage 0 endpoint. We use $\delta_n = 1$ for 256×256 generation and $\delta_n = 2$ for 512×512 generation.

Stage 1 then continues denoising at full resolution from a noisy coarse state:

$$\begin{aligned} x^{s_1} &= (1 - t_{s_1})\widetilde{x}_0 + t_{s_1}\epsilon_1, \qquad \epsilon_1 \sim \mathcal{N}(0, \delta_n^2 I), \\ x^{e_1} &= x_0, \end{aligned} \tag{7}$$

where $\widetilde{x}_0 = \mathcal{U}(\mathcal{D}(x_0))$ is the coarse full-resolution image obtained by downsampling and upsampling x_0, $\mathcal{U}(\cdot)$ denotes $2\times$ nearest-neighbor upsampling, and t_{s_1} specifies the Stage 1 starting noise level. Accordingly, $t_{s_0} = 1$ and $t_{e_1} = 0$ correspond to the pure-noise start of Stage 0 and the clean-image endpoint of Stage 1, respectively. Within each stage $s \in \{0, 1\}$, we introduce a stage-relative timestep $\sigma \in [0, 1]$ and linearly interpolate between the stage start and endpoint:

$$x_\sigma^s = (1 - \sigma)x^{e_s} + \sigma x^{s_s}, \tag{8}$$

where x^{s_s} and x^{e_s} denote the start and endpoint of stage s, respectively; *e.g.*, x^{s_0} and x^{e_0} for Stage 0. The corresponding target velocity is $v^s = \partial x_\sigma^s / \partial \sigma = x^{s_s} - x^{e_s}$.

Stage-Wise Endpoint Prediction. Inspired by the clean-data prediction in JiT (Li & He, 2026), we predict the endpoint of each stage. Specifically, given x_σ^s, PixelMoE predicts the stage endpoint $\hat{x}_\theta^{e_s}(x_\sigma^s, t, c)$, where $t = t_{e_s} + \sigma(t_{s_s} - t_{e_s})$ is the corresponding global timestep used for conditioning the model and c denotes the class condition. The predicted endpoint induces the velocity $v_\theta(x_\sigma^s, t, c) = (x_\sigma^s - \hat{x}_\theta^{e_s}(x_\sigma^s, t, c))/\sigma$. Substituting this parameterization into the velocity-matching objective gives

$$\mathcal{L}_{\text{v-endpoint}} = \mathbb{E}\left[\mathcal{L}_{\text{MSE}}(v_\theta(x_\sigma^s, t, c), v^s)\right] = \mathbb{E}\left[\frac{1}{\sigma^2}\mathcal{L}_{\text{MSE}}(\hat{x}_\theta^{e_s}(x_\sigma^s, t, c), x^{e_s})\right], \tag{9}$$

Method	#Params	#Active	FID↓
Dense-B	223M	223M	4.40
PixelMoE-B-E32A4	461M	225M	**2.46**
Dense-L	622M	622M	2.95
PixelMoE-L-E32A4	1.65B	632M	**1.78**
Dense-H	1.44B	1.44B	2.35
PixelMoE-H-E32A4	3.68B	1.47B	**1.61**

Table 1: **Active parameter scaling: dense *vs.* MoE.**

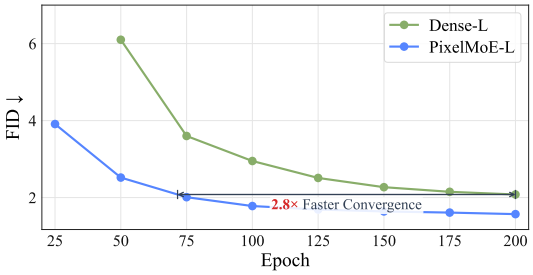


Figure 5: **Convergence comparison between Dense-L and PixelMoE-L.**

where $\mathcal{L}_{\text{MSE}}$ denotes the mean squared error. During both training and sampling, we replace σ in the denominator with $\max(\sigma, \sigma_{\min})$, where $\sigma_{\min} = 0.05$ to avoid numerical instability as $\sigma \to 0$. This velocity-space endpoint objective is the only generation loss used to train the default PixelMoE models; we do not employ representation alignment or perceptual losses.

Coarse-to-Fine Sampling. During sampling, generation proceeds sequentially from Stage 0 to Stage 1. Starting from low-resolution Gaussian noise, PixelMoE first follows the Stage 0 flow to obtain the coarse endpoint $\hat{x}_\theta^{e_0}$. The upsampled Stage 0 endpoint cannot be directly used as the Stage 1 starting state $\hat{x}_\theta^{s_1}$, since both its signal coefficient and noise covariance differ from those of the predefined Stage 1 starting distribution. We therefore rescale the upsampled endpoint and inject an additional Gaussian perturbation:

$$\hat{x}_\theta^{s_1} = \frac{1 - t_{s_1}}{1 - t_{e_0}} \mathcal{U}(\hat{x}_\theta^{e_0}) + \alpha \epsilon', \tag{10}$$

where the scaling factor aligns the signal component with the Stage 1 starting state, while ϵ' follows a block-correlated Gaussian distribution and α is chosen to match the target covariance. The resulting state is constructed to match the predefined Stage 1 statistics while preserving the coarse structure generated in Stage 0. PixelMoE then follows the Stage 1 flow toward the clean full-resolution endpoint. The transition coefficients and block-correlated noise construction are derived in Section B.

Stage-Aware Timestep Sampling. The two stages denoise at different spatial resolutions and cover different portions of the denoising trajectory. We therefore adopt a stage-aware timestep sampling strategy that separately controls stage allocation and stage-relative timestep sampling. Stage 0 covers the global timestep interval $[t_{e_0}, 1]$, while Stage 1 covers $[0, t_{s_1}]$. Specifically, a global $\text{LogitNormal}(\mu_\text{g}, \delta_\text{g})$ distribution determines the sampling probability of each stage, with the probability mass in the overlapping region divided equally between them. After sampling a stage s, we independently sample the stage-relative timestep $\sigma \sim \text{LogitNormal}(\mu_\text{l}, \delta_\text{l})$ and map it to the corresponding global timestep as $t = t_{e_s} + \sigma(t_{s_s} - t_{e_s})$. This formulation decouples stage allocation from stage-relative timestep sampling, allowing the timestep distribution within each stage to be controlled independently of how frequently that stage is selected.

4 Experiments

We conduct all experiments on class-conditional ImageNet-1K generation (Deng et al., 2009).

Training Details. During training, we apply random horizontal flipping with probability 0.5. Models for comparison with previous methods are trained for 320 epochs, while other experiments use 100 epochs unless otherwise specified. For PixelMoE-H, we apply additional regularization after epoch 200. We use a learning rate of 2×10^{-4} with a 5-epoch linear warm-up, followed by a constant schedule with cosine decay over the final 40 epochs. For optimization, we apply Gram Newton-Schulz Muon (gMuon) (Jordan et al., 2024; Zhang et al., 2026) with momentum 0.95 to eligible matrix-valued parameters, while the remaining parameters are optimized with AdamW (Loshchilov & Hutter, 2019) using $(\beta_1, \beta_2) = (0.9, 0.95)$. We employ stage-aware Logit-Normal timestep sampling with $(\mu_\text{g}, \delta_\text{g}) = (0.8, 0.8)$ for stage allocation and $(\mu_\text{l}, \delta_\text{l}) = (0.6, 0.8)$ for stage-relative timestep sampling. Two exponential moving average (EMA) copies with decay rates 0.9996 and 0.9999 are maintained during training. Detailed training configurations are provided in Section E.

Evaluation Protocol. We evaluate generation quality using Fréchet Inception Distance (FID) (Heusel et al., 2017), Inception Score (IS) (Salimans et al., 2016), Precision, and Recall (Kynkäänniemi et al., 2019). We sample 50,000 images with labels uniformly distributed over the ImageNet-1K classes. We use a 50-point global timestep grid with the Heun solver and apply classifier-free guidance (CFG) (Ho & Salimans, 2021) with a CFG interval (Kynkäänniemi et al., 2024). Detailed evaluation settings are provided in Section F.

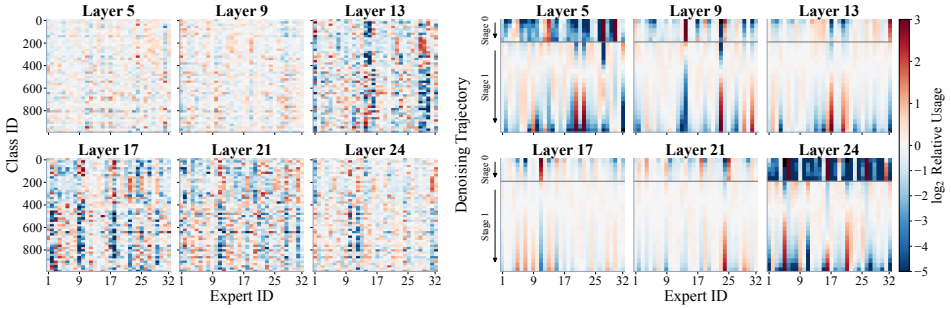
Method	#Params	#Active	FID↓
PixelMoE-L-E8A4	892M	632M	2.28
PixelMoE-L-E16A4	1.14B	632M	2.03
PixelMoE-L-E32A4	1.65B	632M	**1.78**

(a) Effect of scaling the number of experts.

Method	#PS	#Params	#Active	FID↓
Dense-L/16	16	625M	625M	3.15
PixelMoE-L-E32A4/16	16	1.65B	634M	**2.89**
Dense-L/8	8	622M	622M	2.95
PixelMoE-L-E32A4/8	8	1.65B	632M	**1.78**

(b) Computation scaling via token granularity.

Table 2: **Sparse scaling analysis of PixelMoE.** #PS denotes the patch size.



(a) Expert Routing Across Semantic Classes (b) Expert Routing Along the Denoising Trajectory

Figure 6: **Expert routing across semantic classes and along the denoising trajectory.** The left panel shows expert routing across ImageNet-1K classes, while the right panel shows how routing evolves along the denoising trajectory.

4.1 Sparse Scaling Analysis

We first investigate whether sparse MoE scaling can effectively improve pixel-space generation. We study the sparse scaling behavior of PixelMoE from three perspectives: active parameter scaling, expert scaling, and computation scaling via token granularity.

Active Parameter Scaling. We first compare PixelMoE with its dense counterparts across model scales. For controlled comparisons, each dense baseline uses the same hierarchical flow setup and training recipe as its PixelMoE counterpart, with sparse MoE layers replaced by dense FFNs at comparable active parameter counts. As shown in Table 1, PixelMoE consistently achieves lower FID than dense models across Base, Large, and Huge scales. Beyond generation quality, we compare the convergence of Dense-L and PixelMoE-L, extending the training to 200 epochs for this analysis. As shown in Figure 5, PixelMoE-L reaches the FID achieved by Dense-L at 200 epochs after only about 71 epochs, reducing the required training epochs by a factor of approximately 2.8. These results show that sparse MoE scaling improves generation quality while also accelerating convergence at comparable active parameter counts.

Expert Scaling. We next isolate the effect of sparse model capacity by varying the number of routed experts E while fixing the number of activated experts to $A = 4$. As shown in Table 2a, increasing E from 8 to 16 and then 32 improves FID from 2.28 to 2.03 and 1.78, respectively, while keeping the active parameter count fixed at 632M. This shows that PixelMoE effectively benefits from increased total model capacity without increasing the number of active parameters.

Computation Scaling via Token Granularity. We further study how PixelMoE benefits from increased computation by reducing the patch size, which increases the token sequence length while keeping the backbone scale fixed. As shown in Table 2b, PixelMoE consistently outperforms the dense models at both patch sizes, with a larger improvement at finer token granularity. A possible explanation is that smaller patches produce more spatially localized token representations and provide more token-level routing decisions, enabling the router to exploit finer-grained spatial variation.

4.2 Expert Specialization

We analyze the routing behavior of PixelMoE-L-E32A4 on the ImageNet-1K validation set. For selected MoE layers, we collect token-level routing assignments and examine how expert preferences vary across semantic classes and along the denoising trajectory. We report the $\log_2$ ratio of the observed assignment frequency to that under uniform routing, where positive and negative values indicate preferred and suppressed experts, respectively.

As shown in Figure 6, PixelMoE exhibits distinct routing patterns across semantic classes and along the denoising trajectory. Class-dependent routing is relatively weak in shallow layers, becomes clearer in intermediate layers, and is most pronounced around Layers 12–20, where different ImageNet-1K classes favor different subsets of experts. In contrast, routing along the denoising tra-

Method	Epochs	#Params	#Active Params	ImageNet-1K Generation @ 256^2			
				FID↓	IS↑	Precision↑	Recall↑
Latent-space models: Dense							
DiT-XL (Peebles & Xie, 2023)	1400	675M+49M	675M+49M	2.27	278.2	0.83	0.57
SiT-XL (Ma et al., 2024)	1400	675M+49M	675M+49M	2.06	270.3	0.82	0.59
REPA (Yu et al., 2025)	800	675M+49M	675M+49M	1.42	305.7	0.80	0.65
Latent-space models: Sparse MoE							
DiT-MoE-XL (Fei et al., 2024)	–	4.1B+49M	1.5B+49M	1.72	315.7	0.83	0.64
RaceDiT-XL (Yuan et al., 2025)	1450	2.8B+49M	707M+49M	2.06	318.6	0.83	0.60
DiffMoE (Shi et al., 2025)	1400	1.2B+49M	458M+49M	2.13	274.4	0.81	0.60
ProMoE-XL (Wei et al., 2026)	1000	1.6B+49M	675M+49M	2.59	265.6	–	–
Pixel-space models: Dense							
ADM (Dhariwal & Nichol, 2021)	400	554M	554M	4.59	186.7	0.82	0.52
RIN (Jabri et al., 2023)	480	410M	410M	3.42	182.0	–	–
Simple Diffusion (Hoogeboom et al., 2023)	–	2.0B	2.0B	2.44	256.3	–	–
PixelFlow-XL (Chen et al., 2025)	320	677M	677M	1.98	282.1	0.81	0.60
PixNerd (Wang et al., 2026a)	320	700M	700M	1.93	298.0	0.80	0.60
DiP-XL (Chen et al., 2026b)	600	631M	631M	1.79	281.9	0.80	0.63
JiT-G (Li & He, 2026)	600	2.0B	2.0B	1.82	292.6	0.79	0.62
PixelGen-XL (Ma et al., 2026b)	160	676M	676M	1.83	293.6	0.79	0.63
PixelREPA-H (Shin et al., 2026)	600	953M	953M	1.81	317.2	–	–
DeCo-XL (Ma et al., 2026a)	600	682M	682M	1.69	304.0	0.79	0.63
PixelDiT-XL (Yu et al., 2026)	320	797M	797M	1.61	292.7	0.78	0.64
HyperDiT-H (He et al., 2026)	600	952M	952M	1.56	306.5	0.80	0.64
AsymFlow-H (Chen et al., 2026a)	600	953M	953M	1.57	–	–	–
Pixel-space models: Sparse MoE							
JiTMoE (Liu et al., 2025)	200	1.3B	465M	3.10	311.3	–	–
PixelMoE-B-E32A4	320	461M	225M	1.88	298.5	0.79	0.61
PixelMoE-L-E32A4	320	1.65B	632M	1.54	315.9	0.81	0.61
PixelMoE-H-E32A4	320	3.68B	1.47B	**1.49**	306.4	0.80	0.63

Table 3: **Class-conditional generation results on ImageNet-1K at** 256×256**.** For latent-space methods, parameter counts include the generative model and the tokenizer decoder used at inference.



Figure 7: **Qualitative results of PixelMoE.** Samples at 256×256 and 512×512 resolutions.

jectory is already prominent in shallow layers, weakens in intermediate layers, and emerges strongly again in the final layer. The routing pattern also changes noticeably across the transition between the two hierarchical stages. Overall, class-dependent routing specialization is more pronounced in intermediate layers, whereas specialization along the denoising trajectory is stronger in shallow and final layers. Additional analysis in Section G shows that these specialized routing patterns coexist with globally balanced expert utilization.

4.3 Benchmarking against Previous Systems

Table 3 compares PixelMoE with representative latent-space and pixel-space generative systems on class-conditional ImageNet-1K at 256×256 resolution. PixelMoE-B-E32A4 achieves an FID of 1.88 with only 225M active parameters. As total model capacity increases, PixelMoE-L further

Exp.	#Params	#Active	FID↓
w/o Shared Expert	2.00B	632M	2.22
w/ Shared Expert	1.65B	632M	**1.78**

(a) Effect of the shared expert.

Exp.	#Expert Comb.	#Params	#Active	FID↓
PixelMoE-L-E8A1	8	1.65B	632M	1.95
PixelMoE-L-E32A4	35,960	1.65B	632M	**1.78**

(b) Ablation on fine-grained routed experts.

Exp.	#Params	#Active	FID↓
Auxiliary Loss	1.65B	632M	2.05
Balancing Bias	1.65B	632M	**1.78**

(c) Comparison of load-balancing strategies.

Exp.	Type	#Params	#Active	FID↓
Dense-L	Dense	622M	622M	2.95
DiffMoE-L-E32A4	EC	1.67B	653M	2.62
PixelMoE-L-E32A4	TC	1.65B	632M	**1.78**

(d) Token-Choice *vs.* Expert-Choice routing.

Table 4: **Ablation studies of the sparse MoE design in PixelMoE.**

improves the FID to 1.54, while PixelMoE-H reaches 1.49. Notably, these results are achieved without representation alignment (Yu et al., 2025; Shin et al., 2026) or perceptual losses (Ma et al., 2026b). Compared with leading dense pixel-space models, PixelMoE achieves better generation quality. PixelMoE also remains competitive with leading latent-space approaches without relying on a learned VAE. Together with the controlled scaling results in Tables 1 and 2a, these comparisons support LLM-style sparse MoE as an effective scaling paradigm for pixel-space generative modeling. Results at 512×512 resolution are provided in Section H, further showing that the sparse scaling behavior of PixelMoE extends effectively to higher-resolution generation.

We further present qualitative samples in Figure 7. PixelMoE generates coherent images across diverse ImageNet-1K categories, with clear global structures and rich local textures. These samples further illustrate the strong visual fidelity of PixelMoE.

4.4 Ablation Studies

We conduct ablation studies on the key components of the LLM-style MoE design in PixelMoE using the 100-epoch training schedule. Additional ablations are provided in Section I.

The Shared Expert. We first examine the role of the shared expert in Table 4a. Under matched active parameters, introducing a shared expert improves FID from 2.22 to 1.78, while reducing the total parameter count from 2.00B to 1.65B. This suggests that combining shared and routed computation is more effective than allocating the entire active parameter budget to routed experts.

Fine-Grained Routed Experts. We next study the effect of expert granularity under matched total and active parameter counts. As shown in Table 4b, increasing the expert configuration from E8A1 to E32A4 expands the number of possible expert combinations from 8 to 35,960 without increasing either total or active parameters. The corresponding FID improvement from 1.95 to 1.78 indicates that PixelMoE benefits from finer-grained expert composition.

Bias-Based Load-Balancing Strategy. The bias-based load-balancing strategy offers a simple but effective way to regulate expert utilization without introducing an auxiliary loss. As shown in Table 4c, compared with an auxiliary load-balancing loss, the bias-based strategy improves FID from 2.05 to 1.78 with identical total and active parameter counts. Together with the expert-utilization analysis in Section G, the result indicates that PixelMoE can preserve flexible input-adaptive routing while maintaining balanced global expert utilization.

Token-Choice Routing Strategy. We compare Token-Choice and Expert-Choice routing to examine which routing paradigm is better suited to pixel-space diffusion. We implement DiffMoE (Shi et al., 2025), which adopts Expert-Choice routing, using the same pixel-space backbone and training recipe as PixelMoE, while keeping the total and active parameter counts comparable. As shown in Table 4d, Token-Choice routing achieves a substantially lower FID. This suggests that token-centric expert selection better accommodates the heterogeneous pixel-token representations across semantic classes and along the denoising trajectory.

5 Conclusion

This paper presents **PixelMoE**, a pixel-space image generation framework that sparsely scales model capacity by directly adopting an LLM-style Mixture-of-Experts (MoE) architecture. We further incorporate Pixel-Space Hierarchical Flow Matching to progressively increase spatial resolution along the denoising trajectory. Experiments on ImageNet-1K show that PixelMoE consistently outperforms its dense counterparts across model scales and achieves competitive performance relative to leading latent- and pixel-space generative models. Our routing analyses further reveal clear expert specialization across semantic classes and along the denoising trajectory under standard Token-Choice routing, without explicit semantic or timestep routing guidance. These results support LLM-style sparse MoE as an effective scaling paradigm for pixel-space generative modeling.

AI USE STATEMENT

We utilized the large language model GPT-6 Astra to improve the clarity and grammar of the writing in the method and experimental sections. Nevertheless, all technical content, including the conceptual formulation and experiments, remains our own; the large language model was used solely as a tool for linguistic enhancement and stylistic polishing.

REFERENCES

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=B1xsqj09Fm.

Arantxa Casanova, Marlene Careil, Jakob Verbeek, Michal Drozdzal, and Adriana Romero Soriano. Instance-conditioned gan. *Advances in Neural Information Processing Systems*, 34:27517–27529, 2021.

Hansheng Chen, Jan Ackermann, Minseo Kim, Gordon Wetzstein, and Leonidas Guibas. Asymmetric flow models. *arXiv preprint arXiv:2605.12964*, 2026a.

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

Zhennan Chen, Junwei Zhu, Xu Chen, Jiangning Zhang, Xiaobin Hu, Hanzhen Zhao, Chengjie Wang, Jian Yang, and Ying Tai. Dip: Taming diffusion models in pixel space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36136–36146, 2026b.

Kun Cheng, Xiao He, Lei Yu, Zhijun Tu, Mingrui Zhu, Nannan Wang, Xinbo Gao, and Jie Hu. Diff-moe: Diffusion transformer with time-aware and space-adaptive experts. In *Forty-second International Conference on Machine Learning*, 2025.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1280–1297, 2024.

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

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*, pp. 12873–12883, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Zhengcong Fei, Mingyuan Fan, Changqian Yu, Debang Li, and Junshi Huang. Scaling diffusion transformers to 16 billion parameters. *arXiv preprint arXiv:2407.11633*, 2024.

Zhida Feng, Zhenyu Zhang, Xintong Yu, Yewei Fang, Lanxin Li, Xuyi Chen, Yuxiang Lu, Jiaxiang Liu, Weichong Yin, Shikun Feng, et al. Ernie-vilg 2.0: Improving text-to-image diffusion model with knowledge-enhanced mixture-of-denoising-experts. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10135–10145. IEEE, 2023.

Alireza Ganjdanesh, Yan Kang, Yuchen Liu, Richard Zhang, Zhe Lin, and Heng Huang. Mixture of efficient diffusion experts through automatic interval and sub-network selection. In *European Conference on Computer Vision*, pp. 54–71. Springer, 2024.

Zipeng Guo, Lichen Ma, Yu He, Xiaolong Fu, Jingling Fu, Junshi Huang, and Yan Li. Pixelu: A u-shaped transformer for efficient end-to-end pixel diffusion. *arXiv preprint arXiv:2606.27760*, 2026.

Yu He, Lichen Ma, Zipeng Guo, Xinyuan Shan, Jingling Fu, Dong Chen, Junshi Huang, and Yan Li. Hyperdit: Hyper-connected transformers for high-fidelity pixel-space diffusion. *arXiv preprint arXiv:2605.15741*, 2026.

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.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

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

Allan Jabri, David J. Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 14569–14589. PMLR, 23–29 Jul 2023.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *International Conference on Learning Representations*, volume 2025, pp. 23378–23402, 2025.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. *Advances in Neural Information Processing Systems*, 36:65484–65516, 2023.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. *Advances in Neural Information Processing Systems*, 37:122458–122483, 2024.

Yunsung Lee, JinYoung Kim, Hyojun Go, Myeongho Jeong, Shinhyeok Oh, and Seungtaek Choi. Multi-architecture multi-expert diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 13427–13436, 2024.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Yahui Liu, Yang Yue, Jingyuan Zhang, Chenxi Sun, Yang Zhou, Wencong Zeng, Ruiming Tang, and Guorui Zhou. Efficient training of diffusion mixture-of-experts models: A practical recipe. *arXiv preprint arXiv:2512.01252*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

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

Zehong Ma, Longhui Wei, Shuai Wang, Shiliang Zhang, and Qi Tian. Deco: Frequency-decoupled pixel diffusion for end-to-end image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 43600–43610, 2026a.

Zehong Ma, Ruihan Xu, and Shiliang Zhang. Pixelgen: Improving pixel diffusion with perceptual supervision. *arXiv preprint arXiv:2602.02493*, 2026b.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

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.

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.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Minglei Shi, Ziyang Yuan, Haotian Yang, Xintao Wang, Mingwu Zheng, Xin Tao, Wenliang Zhao, Wenzhao Zheng, Jie Zhou, Jiwen Lu, et al. Diffmoe: Dynamic token selection for scalable diffusion transformers. *arXiv preprint arXiv:2503.14487*, 2025.

Jaeyo Shin, Jiwook Kim, and Hyunjung Shim. Representation alignment for just image transformers is not easier than you think. *arXiv preprint arXiv:2603.14366*, 2026.

Haotian Sun, Tao Lei, Bowen Zhang, Yanghao Li, Haoshuo Huang, Ruoming Pang, Bo Dai, and Nan Du. Ec-dit: Scaling diffusion transformers with adaptive expert-choice routing. In *International Conference on Learning Representations*, volume 2025, pp. 72383–72401, 2025.

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

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. In *International Conference on Learning Representations*, volume 2026, pp. 43559–43580, 2026a.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 40633–40642, 2026b.

Yujie Wei, Shiwei Zhang, Hangjie Yuan, Yujin Han, Zhekai Chen, Jiayu Wang, Difan Zou, Xihui Liu, Yu Liu, Hongming Shan, et al. Routing matters in moe: Scaling diffusion transformers with explicit routing guidance. In *International Conference on Learning Representations*, volume 2026, pp. 23381–23405, 2026.

Zeyue Xue, Guanglu Song, Qiushan Guo, Boxiao Liu, Zhuofan Zong, Yu Liu, and Ping Luo. Raphael: Text-to-image generation via large mixture of diffusion paths. *Advances in Neural Information Processing Systems*, 36:41693–41706, 2023.

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

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=DJSZGGZYVi.

Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14273–14282, 2026.

Yike Yuan, Ziyu Wang, Zihao Huang, Defa Zhu, Xun Zhou, Jingyi Yu, and Qiyang Min. Expert race: A flexible routing strategy for scaling diffusion transformer with mixture of experts. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9W3CSuHoFA.

Jack Zhang, Noah Amsel, Berlin Chen, and Tri Dao. Gram newton-schulz, 2026. URL https://dao-ailab.github.io/blog/2026/gram-newton-schulz/.

Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast training of diffusion models with masked transformers. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=vTBjBtGioE.

A Additional Details of Bias-Based Load Balancing

We provide the detailed computation of the expert-load distribution used in the bias-based load-balancing update. For image token i, the assignment to expert e is defined as

$$f^s_{t,i,e} = \begin{cases} \frac{1}{A}, & e \in \mathcal{S}^s_{t,i}, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where $\mathcal{S}^s_{t,i}$ is the set of A routed experts selected for image token i. Averaging the assignments over all N image tokens in the training batch gives the observed expert-load distribution F_e:

$$F_e = \frac{1}{N} \sum_{(s,t,i)\in\mathcal{T}} f^s_{t,i,e}, \tag{12}$$

where $\mathcal{T}$ denotes the image-token indices in the batch. Since each image token contributes a total assignment mass of one, $\sum_{e=1}^{E} F_e = 1$.

The uniform load target is $\mathbf{Q} = (1/E, \ldots, 1/E)$. For the bias update in Section 3.1, RMS normalization is defined as

$$\mathrm{RMS}(\mathbf{z}) = \sqrt{\frac{1}{E}\sum_{e=1}^{E} z_e^2}. \tag{13}$$

Accordingly, the balancing-bias vector is updated as

$$\mathbf{b} \leftarrow \mathbf{b} - \gamma \frac{\mathbf{F} - \mathbf{Q}}{\mathrm{RMS}(\mathbf{F} - \mathbf{Q})}. \tag{14}$$

We use $\gamma = 0.001$ throughout training. The balancing bias is updated without gradient propagation and affects expert selection only; the original routing scores are used to compute the routing weights.

B Scale Transition in Hierarchical Flow Matching Schedule

During sampling, simply upsampling the low-resolution endpoint produced by Stage 0 does not yield a valid Stage 1 starting state, since nearest-neighbor upsampling replicates the noise associated with each low-resolution pixel and consequently introduces local correlations that are absent from the Stage 1 starting distribution used during training. We therefore follow Pyramidal Flow Matching (Jin et al., 2025) and construct a distribution-preserving transition from the Stage 0 endpoint $\hat{x}^{e_0}_\theta$ to the Stage 1 starting state $\hat{x}^{s_1}_\theta$. Specifically, the Stage 0 endpoint and the predefined Stage 1 starting state are given by

$$\begin{aligned} x^{e_0} &= (1 - t_{e_0})\mathcal{D}(x_0) + t_{e_0}\epsilon_0, \\ x^{s_1} &= (1 - t_{s_1})\widetilde{x}_0 + t_{s_1}\epsilon_1, \qquad \widetilde{x}_0 = \mathcal{U}(\mathcal{D}(x_0)), \end{aligned} \tag{15}$$

where $\epsilon_0, \epsilon_1 \sim \mathcal{N}(0, \delta_n^2 I)$.

Assuming an ideal Stage 0 prediction, upsampling its endpoint gives

$$\mathcal{U}(x^{e_0}) = (1 - t_{e_0})\widetilde{x}_0 + t_{e_0}\mathcal{U}(\epsilon_0). \tag{16}$$

To match the signal component of the predefined Stage 1 starting state, we first rescale the upsampled endpoint:

$$\frac{1 - t_{s_1}}{1 - t_{e_0}}\mathcal{U}(x^{e_0}) = (1 - t_{s_1})\widetilde{x}_0 + \beta\,\mathcal{U}(\epsilon_0), \tag{17}$$

where

$$\beta = \frac{1 - t_{s_1}}{1 - t_{e_0}} t_{e_0}. \tag{18}$$

The signal coefficient now matches that of x^{s_1}, while the remaining difference lies in the noise covariance.

For $2\times$ nearest-neighbor upsampling, each low-resolution noise value is copied to a 2×2 block. Therefore, after factoring out the noise variance δ_n^2, the within-block covariance of $\mathcal{U}(\epsilon_0)$ is

$$\Sigma_{\text{up}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{19}$$

In contrast, the noise component $t_{s_1}\epsilon_1$ of the Stage 1 starting state has covariance $t_{s_1}^2\delta_n^2 I_4$, where $I_4 \in \mathbb{R}^{4\times 4}$ denotes the identity matrix over the four pixels within each 2×2 block. We therefore introduce a block-correlated Gaussian perturbation ϵ', sampled independently of the Stage 0 noise, and construct

$$\hat{x}_\theta^{s_1} = \frac{1 - t_{s_1}}{1 - t_{e_0}}\mathcal{U}(\hat{x}_\theta^{e_0}) + \alpha\epsilon', \tag{20}$$

where

$$\epsilon' \sim \mathcal{N}(0, \delta_n^2\Sigma_\rho), \tag{21}$$

and

$$\Sigma_\rho = \begin{bmatrix} 1 & \rho & \rho & \rho \\ \rho & 1 & \rho & \rho \\ \rho & \rho & 1 & \rho \\ \rho & \rho & \rho & 1 \end{bmatrix}. \tag{22}$$

For Σ_ρ to define a valid covariance matrix, it must be positive semidefinite. Its eigenvalues are $1 - \rho$ with multiplicity three and $1 + 3\rho$ with multiplicity one, which requires $-\frac{1}{3} \le \rho \le 1$. Since the additional noise is used to cancel the positive within-block correlations introduced by nearest-neighbor upsampling, we further restrict ρ to $-\frac{1}{3} \le \rho \le 0$.

The actual within-block covariance of the resulting noise component in Equation (20) is

$$\delta_n^2\left(\beta^2\Sigma_{\text{up}} + \alpha^2\Sigma_\rho\right). \tag{23}$$

Matching this covariance to the target covariance $\delta_n^2 t_{s_1}^2 I_4$ of the predefined Stage 1 starting distribution and canceling the common factor δ_n^2 yields

$$\beta^2\Sigma_{\text{up}} + \alpha^2\Sigma_\rho = t_{s_1}^2 I_4, \tag{24}$$

Matching the diagonal and off-diagonal entries gives

$$\beta^2 + \alpha^2 = t_{s_1}^2, \tag{25}$$

$$\beta^2 + \rho\alpha^2 = 0. \tag{26}$$

Solving the two constraints yields

$$\alpha^2 = \frac{t_{s_1}^2}{1 - \rho}, \qquad \beta^2 = -\frac{\rho}{1 - \rho}t_{s_1}^2. \tag{27}$$

Within the admissible range $\rho \in [-1/3, 0]$, a more negative ρ decreases α^2, i.e., the variance coefficient of the additionally injected noise, while increasing the proportion of noise inherited from the upsampled Stage 0 endpoint. To make the transition minimally perturbative to the Stage 0 prediction, we minimize the variance of the newly injected noise subject to the covariance-matching constraint. Following Pyramidal Flow Matching (Jin et al., 2025), we therefore choose the most negative admissible value, $\rho = -1/3$. Substituting it into Equation (27) gives

$$\alpha = \frac{\sqrt{3}}{2}t_{s_1}, \qquad \beta = \frac{1}{2}t_{s_1}. \tag{28}$$

Combining the expression for β with its definition in Equation (18) determines the relation between the two stage-boundary noise levels:

$$\frac{1 - t_{s_1}}{1 - t_{e_0}}t_{e_0} = \frac{t_{s_1}}{2}, \tag{29}$$

which gives

$$t_{e_0} = \frac{t_{s_1}}{2 - t_{s_1}}. \tag{30}$$

Algorithm 1 Sparse MoE Layer in PixelMoE: PyTorch-like Pseudo-code

```
class PixelMoELayer(nn.Module):
    def __init__(
        self, dim, routed_dim, shared_dim,
        E=32, A=4, gamma=1e-3
    ):
        self.router = nn.Linear(dim, E, bias=False)
        self.experts = nn.ModuleList(
            [Expert(routed_dim) for _ in range(E)])
        self.shared_expert = Expert(shared_dim)

        self.router_bias = torch.zeros(E)
        self.A = A
        self.router_scale = math.sqrt(A)
        self.gamma = gamma

    def forward(self, x):
        scores = torch.sigmoid(self.router(x))

        # Balancing bias is used only for expert selection
        expert_ids = torch.topk(
            scores + self.router_bias,
            self.A, dim=-1
        ).indices

        # Normalize the original routing scores
        weights = gather(scores, expert_ids)
        weights = weights / weights.sum(
            dim=-1, keepdim=True)
        weights = self.router_scale * weights

        routed = sparse_expert_forward(
            x, self.experts, expert_ids, weights)

        shared = self.shared_expert(x)
        return routed + shared

    @torch.no_grad()
    def update_router_bias(self, load):
        target = load.mean()
        delta = load - target
        self.router_bias -= (
            self.gamma * delta / rms(delta)
        )
```

Thus, the Stage 0 endpoint and Stage 1 starting noise levels cannot be chosen independently; specifying either one determines the other under the covariance-matching transition.

The resulting cross-stage transition used during sampling is therefore

$$\hat{x}_\theta^{s_1} = \frac{1 - t_{s_1}}{1 - t_{e_0}} \mathcal{U}(\hat{x}_\theta^{e_0}) + \frac{\sqrt{3}}{2} t_{s_1} \epsilon', \qquad \epsilon' \sim \mathcal{N}\big(0, \delta_n^2 \Sigma_{-1/3}\big)\,. \tag{31}$$

This transition preserves the coarse structure predicted by Stage 0 while matching the noise statistics of the predefined Stage 1 starting distribution and minimizing the amount of newly injected noise under the covariance-matching constraint.

Sampling the Correlated Noise. The required block-correlated Gaussian noise can be sampled directly. For each 2×2 spatial block and channel, we draw $\mathbf{z} \sim \mathcal{N}(0, I_4)$ and compute $\bar{z} = \frac{1}{4}\mathbf{1}^\top \mathbf{z}$, where $\mathbf{1} \in \mathbb{R}^4$ denotes the all-ones vector. Then we set

$$\epsilon' = \delta_n \frac{2}{\sqrt{3}} (\mathbf{z} - \bar{z}\mathbf{1})\,. \tag{32}$$

This construction satisfies

$$\mathrm{Cov}(\epsilon') = \delta_n^2 \Sigma_{-1/3}, \tag{33}$$

providing a simple implementation of the correlated re-noising used in the cross-stage transition.

C PSEUDO-CODE OF PIXELMOE

We provide simplified PyTorch-like pseudo-code for the two key components of PixelMoE: the sparse MoE layer in Algorithm 1 and the two-stage pixel-space hierarchical flow matching schedule in Algorithm 2. In Algorithm 2, `heun_solver` evaluates `velocity` at global timesteps t but updates the state using stage-relative increments $\Delta\sigma = \Delta t/(t_{s_s} - t_{e_s})$.

Algorithm 2 Pixel-Space Hierarchical Flow Matching: PyTorch-like Pseudo-code

```
class HierarchicalFlow(nn.Module):
    def __init__(self, model):
        self.model = model  # shared PixelMoE

    def training_loss(self, x, c):
        stage = sample_stage()

        if stage == 0:
            x_low = downsample(x)
            noise = delta_n * torch.randn_like(x_low)

            x_start = noise
            x_end = (1 - t_e0) * x_low + t_e0 * noise
            t_start, t_end = 1.0, t_e0

        else:
            x_coarse = upsample(downsample(x))
            noise = delta_n * torch.randn_like(x)

            x_start = (1 - t_s1) * x_coarse + t_s1 * noise
            x_end = x
            t_start, t_end = t_s1, 0.0

        # Stage-relative timestep
        sigma = sample_timestep(stage)
        x_sigma = sigma * x_start + (1 - sigma) * x_end
        # Corresponding global timestep
        t = t_end + sigma * (t_start - t_end)

        # Stage-wise endpoint prediction
        x_end_pred = self.model(x_sigma, t, c)

        sigma_safe = sigma.clamp_min(sigma_min)
        loss = ((x_end_pred - x_end) / sigma_safe) ** 2
        return loss.mean()

    def velocity(self, x, t, c, stage):
        if stage == 0:
            t_start, t_end = 1.0, t_e0
        else:
            t_start, t_end = t_s1, 0.0
        # Convert global timestep to stage-relative timestep
        sigma = (t - t_end) / (t_start - t_end)

        # Convert endpoint prediction to velocity
        x_end_pred = self.model(x, t, c)
        sigma_safe = sigma.clamp_min(sigma_min)
        return (x - x_end_pred) / sigma_safe

    @torch.no_grad()
    def sample(self, c):
        # Stage 0: low-resolution generation
        x = delta_n * torch.randn(low_res_shape)
        x = heun_solver(self.velocity, x, c, stage=0)

        # Distribution-preserving stage transition
        x = stage_transition(x)

        # Stage 1: full-resolution refinement
        x = heun_solver(self.velocity, x, c, stage=1)

        return x
```

D Model Configuration

We implement PixelMoE at Base, Large, and Huge scales, with the detailed configurations summarized in Table 5. For 256×256 generation, all models use a patch size of 8; we use a patch size of 16 at 512×512. All models adopt the E32A4 MoE configuration unless otherwise specified, where each image token activates 4 experts from a pool of 32 routed experts. As the backbone scales from Base to Huge, we increase its depth and hidden dimension together with the capacity of both routed and shared experts. A wide DDT head is appended to each backbone for stage-wise endpoint prediction, using 2, 2, and 4 blocks for the Base, Large, and Huge models, respectively. Across all scales, the first four backbone blocks remain dense, while the remaining backbone blocks adopt sparse MoE layers; the DDT head retains dense FFNs.

Active Parameter Counting. In MoE blocks, image tokens are processed by the sparse MoE FFN, while class tokens use a separate dense FFN. We report the average active parameter count per token, weighted according to the respective numbers of image and class tokens during training.

E Additional Training Details

We summarize the detailed training configurations of PixelMoE in Table 6. Unless otherwise specified, the same training recipe is used across model scales. For 512×512 generation, we increase the patch size from 8 to 16 and the noise scale from 1.0 to 2.0, while keeping the remaining optimization, hierarchical flow, and MoE settings unchanged.

Controlled Dense Baselines. For controlled comparisons, each dense baseline follows the same backbone configuration, hierarchical flow schedule, DDT prediction head, optimizer, timestep sampling, and training recipe as the corresponding PixelMoE model. The sparse MoE layers are replaced by dense FFNs with comparable active parameter counts, while all other architectural and training settings are kept unchanged.

Regularization. For the final PixelMoE-H models, we introduce additional regularization after epoch 200. Following LlamaGen (Sun et al., 2024), we use a crop range of 1.1, where each training image is first resized to $1.1\times$ the target resolution and then randomly cropped to the target size. We additionally apply dropout with a rate of 0.2 to the output of each self-attention module and feed-forward network. These modifications increase the diversity of training views and provide stronger regularization during the later stage of training.

Optimization. We use gMuon (Zhang et al., 2026) for eligible matrix-valued parameters, including the routed-expert matrices, while AdaLN layers, the final prediction layer, timestep and class embedders, and the remaining parameters are optimized with AdamW (Loshchilov & Hutter, 2019). We use a learning rate of 2×10^{-4} with a global batch size of 1024. After a 5-epoch warm-up, we keep the learning rate constant and apply cosine decay to 2×10^{-5} over the final 40 epochs. We train with `bfloat16` mixed precision and maintain two EMA copies with decay rates 0.9996 and 0.9999.

Hierarchical Flow Training. We use the same two-stage pixel-space hierarchical flow matching schedule for both resolutions, with Stage 0 denoising at half spatial resolution and Stage 1 at full resolution. We set the Stage 0 endpoint to $t_{e_0} = 0.75$ and use $\rho = -1/3$ for the noise correlation. Under the cross-stage transition derived in Section B, these settings determine the Stage 1 starting noise level as $t_{s_1} = 6/7$. The relative sampling frequency of the two stages is determined by a global Logit-Normal distribution with $(\mu_{\mathrm{g}}, \delta_{\mathrm{g}}) = (0.8, 0.8)$, with the probability mass in the overlapping stage interval divided equally between the two stages. The stage-relative timestep is then independently sampled from a Logit-Normal distribution with $(\mu_{\mathrm{l}}, \delta_{\mathrm{l}}) = (0.6, 0.8)$. Under the resulting stage intervals, this yields sampling probabilities $p(s = 0) \approx 0.231$ and $p(s = 1) \approx 0.769$.

Stage-wise Endpoint Prediction Objective. PixelMoE directly predicts the endpoint of each stage while using the velocity-space flow matching loss introduced in Section 3.2. For numerical stability near the stage endpoint, we clamp the inverse-square weighting using $\sigma_{\min} = 0.05$. The

Model	Depth	Hidden Dim.	Heads	DDT Depth	DDT Dim.	Routed Dim.	Shared Dim.	#Dense
PixelMoE-B-E32A4	12	768	12	2	1536	384	632	4
PixelMoE-L-E32A4	24	1024	16	2	2048	512	836	4
PixelMoE-H-E32A4	32	1280	16	4	2560	640	1048	4

Table 5: **PixelMoE model configurations.** DDT Dim. represents the hidden state dimension in the DDT head. Routed Dim. and Shared Dim. denote the intermediate dimensions of the routed-expert and shared-expert feed-forward networks (FFNs), respectively. #Dense denotes the number of dense blocks in the PixelMoE backbone.

implemented objective is

$$\mathcal{L}_{\text{v-endpoint}} = \mathbb{E}\left[\frac{\mathcal{L}_{\text{MSE}}\left(\hat{x}_\theta^{e_s}, x^{e_s}\right)}{\max(\sigma, \sigma_{\min})^2}\right], \tag{34}$$

where the expectation is taken over images, sampled stages, stage-relative timesteps, and Gaussian noise. The inverse-square weighting is not an additional loss-reweighting heuristic. Before applying the numerical clamp, the endpoint parameterization gives

$$v_\theta - v^s = \frac{x^{e_s} - \hat{x}_\theta^{e_s}}{\sigma}, \tag{35}$$

so velocity-space MSE directly yields the $1/\sigma^2$ weighting on endpoint-prediction error. The clamp at $\sigma_{\min}$ only prevents this factor from becoming singular near the stage endpoint.

MoE Training. The router operates in FP32 and converts the routing logits into sigmoid scores. For each image token, we select the top-A experts according to the bias-adjusted scores, normalize the original sigmoid scores over the selected experts, and multiply the resulting routing weights by a scale factor of $\sqrt{A}$ before expert aggregation. The balancing bias only affects expert selection and is not used when computing the expert routing weights. After each optimizer update, expert loads accumulated during the corresponding forward passes are used to update the balancing bias with an RMS-normalized correction and a step size of 0.001. No auxiliary load-balancing objective is used for the default PixelMoE models.

Auxiliary Load-Balancing Baseline. For the auxiliary load-balancing loss baseline in Table 4c, we replace the bias-based balancing strategy with the Token-Choice load-balancing loss (Fedus et al., 2022; Liu et al., 2024). Specifically, let $r_{i,e}$ denote the sigmoid routing score of image token i for expert e, and let $\mathcal{S}_i$ denote its selected top-A experts. We first normalize the routing scores over all experts as

$$p_{i,e} = \frac{r_{i,e}}{\sum_{j=1}^{E} r_{i,j}}. \tag{36}$$

For each expert, we define the mean routing probability and assignment fraction as

$$P_e = \frac{1}{N}\sum_{i=1}^{N} p_{i,e}, \qquad F_e = \frac{1}{NA}\sum_{i=1}^{N} \mathbf{1}[e \in \mathcal{S}_i], \tag{37}$$

where N is the number of image tokens. The load-balancing loss for the ℓ-th MoE layer is

$$\mathcal{L}_{\text{bal}}^{(\ell)} = E\sum_{e=1}^{E} \text{sg}(F_e) P_e, \tag{38}$$

where $\text{sg}(\cdot)$ denotes stop-gradient. We average this loss across all MoE layers and add it to the training objective:

$$\mathcal{L}_{\text{aux}} = \lambda_{\text{aux}} \cdot \frac{1}{L_{\text{MoE}}} \sum_{\ell=1}^{L_{\text{MoE}}} \mathcal{L}_{\text{bal}}^{(\ell)}, \tag{39}$$

where L_{MoE} denotes the number of MoE layers. Following standard practice in prior MoE models (Fedus et al., 2022; Dai et al., 2024), we set $\lambda_{\text{aux}} = 0.01$. For this baseline, the training objective is

$$\mathcal{L} = \mathcal{L}_{\text{v-endpoint}} + \mathcal{L}_{\text{aux}}. \tag{40}$$

	256×256			512×512		
Configuration	Base	Large	Huge	Base	Large	Huge
Input and training						
Training resolution	256^2	256^2	256^2	512^2	512^2	512^2
Patch size	8	8	8	16	16	16
Noise scale	1.0	1.0	1.0	2.0	2.0	2.0
Crop range	–	–	$1.1^\dagger$	–	–	$1.1^\dagger$
Dropout	0	0	$0.2^\dagger$	0	0	$0.2^\dagger$
Label dropout	0.1					
Epochs	100 (default for scaling/ablation/routing), 320 (benchmarking)					
Global batch size	1024					
Random horizontal flip	0.5					
Optimization						
Optimizer	gMuon + AdamW					
Learning rate	2×10^{-4}					
Warm-up epochs	5					
LR schedule	Constant + cosine tail					
Cosine decay epochs	40					
Minimum learning rate	2×10^{-5}					
Weight decay	0					
gMuon momentum	0.95					
AdamW (β_1, β_2)	$(0.9, 0.95)$					
Precision	bfloat16					
EMA decay	0.9999, 0.9996					
Hierarchical flow training						
Number of stages	2					
Stage resolutions	$128^2, 256^2$	$128^2, 256^2$	$128^2, 256^2$	$256^2, 512^2$	$256^2, 512^2$	$256^2, 512^2$
Stage 0 endpoint t_{e_0}	0.75					
Stage 1 start t_{s_1}	$6/7$					
Noise correlation ρ	$-1/3$					
Global timestep distribution	Logit-Normal $(\mu_g = 0.8, \delta_g = 0.8)$					
Stage-relative distribution	Logit-Normal $(\mu_l = 0.6, \delta_l = 0.8)$					
Endpoint-loss clamp $\sigma_{\min}$	0.05					
MoE training						
Router score	Sigmoid					
Router precision	FP32					
Routing scale	$\sqrt{A}$					
Load balancing	Bias-based					
Bias update rate	0.001					
Bias update rule	RMS-normalized					

Table 6: **Training configurations of PixelMoE.** For 512×512 generation, we increase the patch size from 8 to 16 and the Gaussian noise scale from 1.0 to 2.0. † For the final PixelMoE-H models, LlamaGen-style crop augmentation with a crop range of 1.1 and dropout with a rate of 0.2 are applied after epoch 200; the 100-epoch controlled experiments use the standard preprocessing without these additional regularization measures.

F Evaluation Details

We follow the evaluation setup of JiT (Li & He, 2026) for FID and IS, including its ImageNet-1K reference statistics, and use the ADM (Dhariwal & Nichol, 2021) implementation for Precision and Recall. We summarize our sampling protocol in Table 7. During sampling, we construct a 50-point uniform global timestep grid, include the stage boundaries, and use the grid points within

Configuration	Setting
Sampling solver	Heun
Global timestep grid	50
Global grid spacing	Uniform
Timestep shifting	None
CFG scale	Swept per model
CFG interval	Swept per model
EMA candidates	0.9996, 0.9999
Selection samples	$10,000$
Final evaluation samples	$50,000$
Class sampling	Uniform over $1,000$ classes
Metrics	FID, IS, Precision, Recall

Table 7: **Evaluation configurations.** CFG scale, CFG interval, and EMA decay are jointly selected for each model according to FID-10K, and the selected configuration is fixed for the final FID-50K evaluation.

the timestep interval of each stage. For class-conditional ImageNet-1K generation, class labels are uniformly distributed over the $1,000$ classes across distributed workers.

Classifier-Free Guidance. We apply classifier-free guidance (Ho & Salimans, 2021) as

$$v_{\text{cfg}}(x_\sigma^s, t, c) = v_\theta(x_\sigma^s, t, \phi) + w(t)\left(v_\theta(x_\sigma^s, t, c) - v_\theta(x_\sigma^s, t, \phi)\right), \tag{41}$$

where c and ϕ denote the class and null conditions, respectively, and $w(t)$ is controlled by the CFG scale w and the CFG interval (Kynkäänniemi et al., 2024). For an interval $[t_{\min}, t_{\max}]$, we define

$$w(t) = \begin{cases} w, & t_{\min} < t < t_{\max}, \\ 1, & \text{otherwise.} \end{cases} \tag{42}$$

Thus, CFG is applied only within the selected portion of the sampling trajectory. Outside this interval, sampling uses the standard conditional prediction without CFG amplification.

Model Selection. For every evaluated model, we jointly sweep the CFG scale, CFG interval, and the two EMA checkpoints. Each candidate configuration is first evaluated with 10,000 samples, and the configuration achieving the lowest FID-10K score is selected. We then regenerate 50,000 samples using the selected configuration and report all final generation metrics on this sample set.

G Global Expert Utilization

The analysis in Figure 6 reveals clear expert specialization across semantic classes and along the denoising trajectory. However, such structured routing does not necessarily imply balanced expert utilization. We therefore examine the routed expert loads across different MoE layers of PixelMoE-L-E32A4. Specifically, for each layer, we accumulate the expert assignments over the ImageNet-1K validation set and report the resulting expert-load distribution. As shown in Figure 8, the routed expert loads remain close to the uniform distribution across all analyzed layers. This indicates that the bias-based load-balancing strategy effectively maintains globally balanced expert utilization while preserving input-adaptive expert specialization. Importantly, the observed routing patterns across semantic classes and along the denoising trajectory do not simply reflect persistent over-utilization of a small subset of experts.

H ImageNet-1K Generation at 512×512

We further evaluate whether the sparse scaling behavior of PixelMoE extends to higher-resolution pixel-space generation. Table 8 reports class-conditional ImageNet-1K results at 512×512, where all PixelMoE models retain the same E32A4 sparse MoE design used at 256×256. Despite the

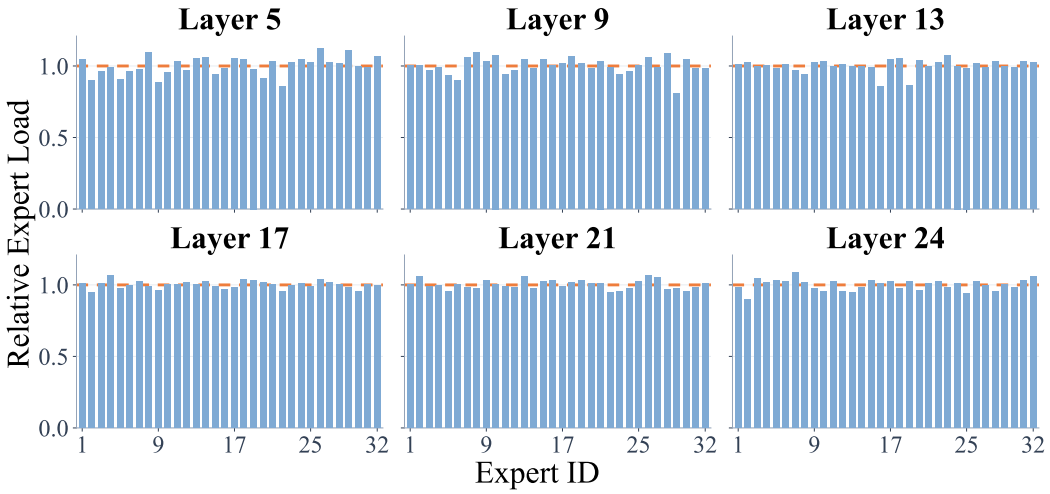


Figure 8: **Global expert utilization across layers.** The routed expert loads of PixelMoE-L-E32A4 remain close to the uniform distribution across the analyzed MoE layers. The dashed line denotes uniform utilization.

Method	Epochs	#Params	#Active Params	**ImageNet-1K Generation @ 512^2**			
				FID↓	IS↑	Precision↑	Recall↑
Latent-space models: Dense							
DiT-XL (Peebles & Xie, 2023)	1400	675M+49M	675M+49M	3.04	240.8	0.84	0.54
SiT-XL (Ma et al., 2024)	1400	675M+49M	675M+49M	2.62	252.2	0.84	0.57
MaskDiT (Zheng et al., 2024)	1600	675M+49M	675M+49M	2.50	256.2	0.83	0.56
REPA (Yu et al., 2025)	800	675M+49M	675M+49M	2.08	274.6	0.83	0.58
Latent-space models: Sparse MoE							
DiT-MoE-XL (Fei et al., 2024)	–	4.1B+49M	1.5B+49M	2.30	298.4	0.85	0.57
Pixel-space models: Dense							
ADM (Dhariwal & Nichol, 2021)	400	554M	554M	3.85	221.7	0.84	0.53
RIN (Jabri et al., 2023)	480	410M	410M	3.95	216.0	–	–
Simple Diffusion (Hoogeboom et al., 2023)	–	2.0B	2.0B	3.02	248.7	–	–
PixNerd-XL (Wang et al., 2026a)	320	700M	700M	2.84	245.6	0.80	0.59
DiP-XL (Chen et al., 2026b)	600	631M	631M	2.31	291.7	0.84	0.58
PixelDiT-XL (Yu et al., 2026)	320	797M	797M	1.81	278.6	0.78	0.67
JiT-G (Li & He, 2026)	600	2.0B	2.0B	1.78	306.8	–	–
DeCo-XL (Ma et al., 2026a)	340	682M	682M	2.22	290.0	0.80	0.60
Pixel-space models: Sparse MoE (Ours)							
PixelMoE-B-E32A4	320	463M	227M	2.05	317.3	0.80	0.61
PixelMoE-L-E32A4	320	1.65B	634M	1.58	330.5	0.81	0.61
PixelMoE-H-E32A4	320	3.68B	1.47B	**1.56**	323.3	0.80	0.62

Table 8: **Class-conditional generation results on ImageNet-1K at** 512×512**.** For latent-space methods, parameter counts include the generative model and the tokenizer decoder used at inference.

substantially increased dimensionality of the pixel-space generation problem, PixelMoE continues to achieve strong generation performance across model scales. PixelMoE-B-E32A4 achieves an FID of 2.05, while PixelMoE-L and PixelMoE-H further improve the FID to 1.58 and 1.56, respectively. PixelMoE remains competitive with representative dense pixel-space and latent-space generative models at this higher resolution. These results indicate that the sparse scaling behavior observed at 256×256 extends effectively to higher-resolution pixel-space generation.

Routing Strategy	#Params	#Active	FID↓
+ Timestep Routing	1.65B	633M	2.10
+ Semantic Routing	1.65B	632M	1.85
PixelMoE-L-E32A4	1.65B	632M	**1.78**

(a) Explicit routing guidance.

Generation Schedule	#Params	#Active	FID↓
Single-stage flow	461M	225M	2.54
Hierarchical flow	461M	225M	**2.46**

(b) Hierarchical flow schedule.

Timestep Sampling	#Params	#Active	FID↓
Global Distribution	461M	225M	2.65
Stage-Aware Sampling	461M	225M	**2.46**

(c) Timestep sampling strategy.

Prediction Head	#Params	#Active	FID↓
w/o DDT Head	369M	133M	3.76
w/ DDT Head	461M	225M	**2.46**

(d) DDT prediction head.

Table 9: **Additional ablation studies of PixelMoE.** Explicit routing guidance is evaluated on PixelMoE-L-E32A4, while the remaining ablations are conducted on PixelMoE-B-E32A4. Unless otherwise specified, models are trained for 100 epochs.

I More Ablation Studies

We provide additional ablations on explicit routing guidance and several components of PixelMoE. Explicit routing guidance is evaluated on PixelMoE-L-E32A4, while the hierarchical flow schedule, stage-aware timestep sampling, and DDT prediction head are evaluated on PixelMoE-B-E32A4. Unless otherwise specified, all experiments follow the same 100-epoch training protocol.

Explicit Routing Guidance. A natural question is whether expert routing requires explicit timestep or semantic information. For timestep-aware routing, we provide the timestep embedding directly to the router together with the token hidden states; for semantic-aware routing, we follow the semantic routing design of ProMoE (Wei et al., 2026). As shown in Table 9a, neither variant improves upon the default PixelMoE router, which makes routing decisions solely from token hidden states. Together with the expert specialization observed across semantic classes and along the denoising trajectory in Figure 6, these results suggest that token hidden states in pixel-space diffusion already contain rich timestep and semantic cues that can support effective routing without explicit guidance.

Hierarchical Flow Schedule. The pixel-space hierarchical flow schedule progressively changes the spatial granularity along the denoising trajectory. We compare it with a single-stage counterpart that performs the entire denoising process at full resolution. For a fair comparison, we adjust the number of training iterations such that both variants process the same total number of image tokens. As shown in Table 9b, hierarchical flow improves FID from 2.54 to 2.46. This result shows the benefit of adapting spatial granularity along the denoising trajectory by using coarser representations for early denoising and finer representations for later refinement.

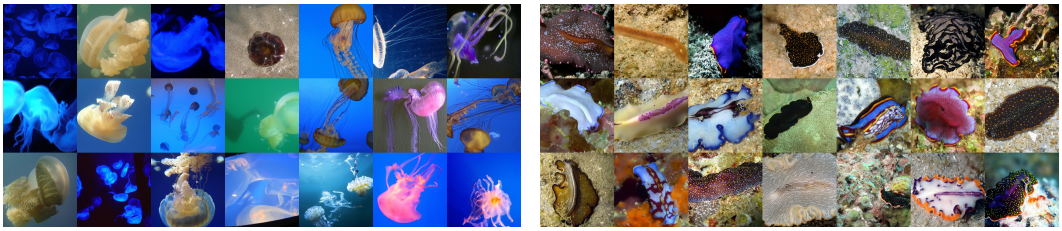
Stage-Aware Timestep Sampling. The two hierarchical stages denoise at different spatial resolutions, motivating stage-aware timestep sampling. The baseline uses a single global Logit-Normal distribution $(\mu_{\mathrm{g}}, \delta_{\mathrm{g}}) = (0.8, 0.8)$ to sample timesteps over the complete denoising trajectory. In contrast, our default strategy uses the same global distribution for stage allocation and independently samples the stage-relative timestep σ within each stage from a Logit-Normal distribution with $(\mu_{\mathrm{l}}, \delta_{\mathrm{l}}) = (0.6, 0.8)$. As shown in Table 9c, this stage-aware sampling strategy improves FID from 2.65 to 2.46, allowing the timestep distribution within each stage to be controlled independently of how frequently that stage is sampled.

DDT Prediction Head. We further ablate the Decoupled Diffusion Transformer (DDT) prediction head (Wang et al., 2026b) on PixelMoE-B-E32A4. Without DDT, the backbone features are mapped directly to the high-dimensional pixel-space prediction, whereas the default model introduces a wider prediction pathway after the backbone. As shown in Table 9d, the DDT head improves FID from 3.76 to 2.46. This result supports the use of the DDT prediction head in our final configuration.



Class 012: House finch, Linnet, Carpodacus mexicanus

Class 042: Agama



Class 107: Jellyfish

Class 110: Flatworm, Platyhelminth



Class 130: Flamingo

Class 288: Leopard, Panthera pardus



Class 349: Bighorn, Bighorn sheep, Cimarron, Rocky Mountain bighorn

Class 425: Barn



Class 453: Bookcase

Class 495: China cabinet, China closet

Figure 9: ***Uncurated* samples of PixelMoE-H-E32A4 on ImageNet-1K at** 256×256**.** Samples are generated conditioned on the specified classes using the same sampling configuration as the reported quantitative results.

J More Visualization Results

We provide additional *uncurated* samples of PixelMoE-H-E32A4 on ImageNet-1K. Figures 9 and 10 show samples at 256×256, while Figure 11 presents samples at 512×512. All samples are generated conditioned on the specified classes using the same sampling settings as the reported quantitative results.



Class 658: Mitten



Class 718: Pier



Class 725: Pitcher, Ewer



Class 779: School bus



Class 829: Streetcar, Tram, Tramcar, Trolley, Trolley car



Class 873: Triumphal arch



Class 911: Wool, Woolen, Woollen



Class 927: Trifle



Class 946: Cardoon



Class 975: Lakeside, Lakeshore

Figure 10: ***Uncurated*** **samples of PixelMoE-H-E32A4 on ImageNet-1K at** 256×256**.** Samples are generated conditioned on the specified classes using the same sampling configuration as the reported quantitative results.



Class 014: Indigo bunting, Indigo finch, Indigo bird, Passerina cyanea



Class 117: Chambered nautilus, Pearly nautilus, Nautilus



Class 309: Bee



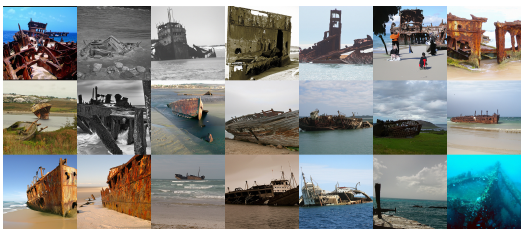
Class 458: Brass, Memorial tablet, Plaque



Class 724: Pirate, Pirate ship



Class 853: Thatch, Thatched roof



Class 913: Wreck



Class 989: Hip, Rose hip, Rosehip

Figure 11: ***Uncurated*** **samples of PixelMoE-H-E32A4 on ImageNet-1K at** 512×512**.** Samples are generated conditioned on the specified classes using the same sampling configuration as the reported quantitative results.