JUST IMAGE TRANSFORMERS WITH LATE-STAGE PIXEL REFINEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work on latent-free diffusion transformers has demonstrated the feasibility of training image generators without a latent VAE. This avoids compression artifacts from the VAE and simplifies the training pipeline. In particular, JiT showed that even a standard transformer can be trained to generate images when the model is trained to predict the final image (x-prediction). Even though JiT unlocks training efficiency comparable to latent models, it still suffers from reduced image quality likely arising from modeling the image as a collection of large patches. In this work, we revisit JiT and show that adding simple convolutional pixel-pathway layers in a post-training phase significantly improves generation quality. Moreover, we show that the added pixel-pathway does not need to be active in the early denoising stages, and can be activated as late as in the last 20% of the denoising process. Our experiments on ImageNet@256 show improvements compared to the original JiT results, achieving an FID of 1.78, without using additional losses.

1 INTRODUCTION

Recent advances (Li & He, 2026; Yu et al., 2026; Chen et al., 2026c; Ma et al., 2026c; Chen et al., 2026b; Guo et al., 2026; You et al., 2026) in latent-free generation avoid the bottleneck of latent-based methods by training diffusion models directly in pixel space. Such pixel-space diffusion transformers (pDiTs) take an RGB image as input and predict a denoising step in pixel space. Li & He (2026) observe that the v-prediction objective commonly used to train diffusion transformers requires the network to predict off-manifold targets because of the added noise component. They instead propose predicting the final image (*x-prediction*), thereby keeping the prediction targets on the manifold of real images. Their experiments show that, where v-prediction fails with large patches, x-prediction succeeds with patches of up to 32×32 pixels. Their model is a standard image transformer that divides an image into large, non-overlapping patches, hence the name Just Image Transformer (JiT).

Although these results show that simple large-patch transformers can be trained directly in pixel space, we find that JiT outputs lack sharpness and show subtle patch-boundary artifacts (see Figs. 1 and 2). We hypothesize that these issues mainly arise from the architectural choices. In fact, JiT independently decodes final patch tokens into non-overlapping pixel patches. This requires consistency across patch boundaries to be established through global patch-level attention. Moreover, the output head has no direct access to the noisy input $\mathbf{x}_t$, so recoverable fine detail must pass through the input bottleneck and the entire transformer backbone. These architectural choices place the burden of resolving local dependencies, such as edge continuation and texture alignment, on a large-patch transformer, which is inefficient use of global attention and transformer capacity. This motivates using dedicated local processing that communicates across patch boundaries.

In this work, we study the architectural and training requirements for pixel-space diffusion transformers to learn fine, spatially coherent details. We augment the patch-level transformer with a pixel-level pathway and study its late activation. We also revisit the timestep-sampling and loss-weighting strategies used during training. Our modifications can be added to pretrained JiT checkpoints through fine-tuning, avoiding the cost of training the full model from scratch.

Our contributions are as follows:

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

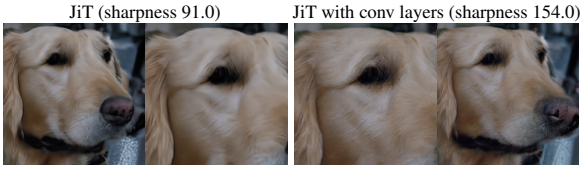


Figure 1: Loss of sharpness in original JiT (left) vs. adding our pixel pathway (right). Measured sharpness (Laplacian variance) reported above.

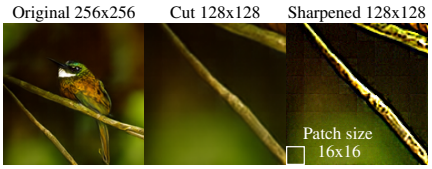


Figure 2: Patch boundaries in JiT-L. The right image is sharpened to accentuate the patch borders.

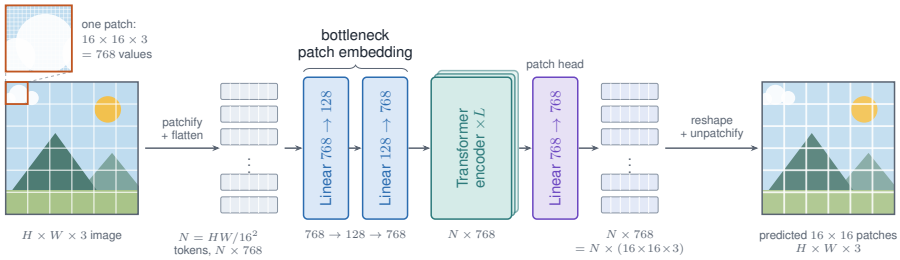


Figure 3: The architecture used by Li & He (2026) is an image transformer with an input bottleneck.

1. **Post-training refinement.** We improve JiT models by adding a pixel-level pathway consisting of only a few convolutional blocks on top of the big-patch transformer. Unlike previous work, we do so in a post-training manner, showing it's possible to pre-train just a transformer backbone using an x-prediction objective and add pixel refinement in a shorter fine-tuning stage.
2. **Efficient late-stage processing.** We show that, when combined with x-prediction, the pixel-level pathway is not necessary throughout the entire denoising process. Consequently, we propose to use the pixel pathway only in the last $\leq 25\% - 50\%$ of denoising steps and show this has minimal to no impact on image quality compared to full-time application. This result further demonstrates the promise of x-prediction for pixel-space modeling while reducing generation cost compared to previously proposed methods with a dedicated pixel pathway.
3. **Thorough analysis.** We revisit the training protocol, showing that the timestep distribution during training has an effect on learning fine details and the timesteps near the clean endpoint benefit from better coverage. We also experiment with different head configurations and revisit the generation procedure of JiT, proposing a correction for the final denoising steps.

2 Approach

In this work, we investigate the addition of a **small pixel-level prediction head** that enables communication across the patch boundaries of JiT's original transformer. Note that while the general idea of using a dedicated pixel-level pathway has been explored in recent work, some of the existing methods do not enable information sharing across big patches in the pixel-level pathway (DiP (Chen et al., 2026c)), or still use global attention over recombined big-patches (PixelDiT (Yu et al., 2026)).

In addition to model modification, we propose a **two-stage training process**. First, the large-patch backbone is pre-trained as described by Li & He (2026). Then, in an additional training phase, the model is adapted and further fine-tuned with an emphasis on recovering high-frequency details in images that plain JiT wasn't able to learn. We believe this enables more efficient training, since JiT training is the most efficient among recent pDiT's (Jiang et al., 2026) as it doesn't require running an additional compute-heavy pixel head throughout the entire training.

Below, we first give an overview of JiT (Sec. 2.1) and discuss the simple convolution pixel head that operates on small 2×2 patches (Sec. 2.2). Then, we propose a more efficient generation procedure that saves compute by running the pixel head only in later denoising stages (Sec. 2.3).

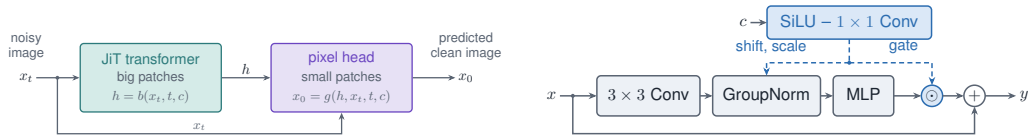


Figure 4: Our model architecture (left) and ConvNeXt-inspired pixel-head block architecture (right). Note the similarity of the ConvNeXt block to a transformer block.

2.1 Preliminaries: JiT

Li & He (2026) use a standard transformer architecture with a bottleneck at the input. Given an RGB image $x \in \mathbb{R}^{H \times W \times 3}$, it is first rearranged into a collection of large patches of size s_p (e.g. patches of 16×16 pixels): $p \in \mathbb{R}^{h \times w \times (3 * s_p^2)}$, where $h = H/s_p$ and $w = W/s_p$. Each patch is then transformed using a bottleneck layer to a patch token. The patch tokens are then processed by a transformer and the output image is produced simply by mapping updated patch features back to pixel space using a linear layer. This transformer has a similar size and computational complexity as its latent counterparts. JiT's model is illustrated in Fig. 3.

Li & He (2026) show that when the model is made to predict $v = x - \epsilon$ (v-prediction for rectified flows), training fails with large patch sizes. This is explained by the fact that the velocity target v contains a noise component and does not lie on the hypothesized low-dimensional manifold of natural images. However, they found that requiring the model predict x (x-prediction) instead enables the use of large patches (up to $s_p = 64$ for 1024×1024 images was tested). They convert x-predictions to velocities, $\hat{\mathbf{v}}_\theta = (\hat{\mathbf{x}}_\theta - \mathbf{x}_t)/(1-t)$, and minimize $\mathcal{L}_v = \mathbb{E}_{x,\epsilon,t}[\|\hat{\mathbf{v}}_\theta - v\|_2^2]$.

2.2 Convolutional Pixel Head

We replace the original linear output layer of the JiT transformer with a stack of ConvNeXt-inspired (Liu et al., 2022) blocks. Given the noisy image $\mathbf{x}_t$, it first passes through the JiT big-patch transformer resulting in a hidden representation

$$h = b(\mathbf{x}_t, t, c) \ , \tag{1}$$

where c is the condition embedding. Subsequently, the pixel head is used to obtain a clean-image estimate,

$$\hat{\mathbf{x}}_\theta = g(h, \mathbf{x}_t, t, c) \ . \tag{2}$$

Note that the noisy image $\mathbf{x}_t$ is also passed directly to the pixel head, along with the backbone features h.

The pixel head consists of a stack of residual ConvNeXt-inspired blocks (see Fig. 4), each containing a dense 3×3 convolution and two 1×1 convolutions. The noisy image $\mathbf{x}_t$ is first split into small non-overlapping patches (2×2), and embedded before being fed to the stack of convolutional blocks. We use small patches (rather than pure pixels like previous work) for increased computational efficiency. Note that unlike previous work that used convolutions (DiP's UNet), our pixel head operates over the entire image and is not constrained to a single big patch. This enables low-level communication across large-patch boundaries.

Why convolutions? In early experiments, we explored more advanced architectures for the pixel head like using sliding window attention with small windows. Both convolutions and sliding window attention implement the inductive bias that local texture matching and patch boundary continuity should be ensured only using local low-level features. While sliding window attention was effective as well, we found it to be significantly more expensive and thus omitted it from further experiments.

Conditioning on backbone features. To condition the pixel head on the backbone's features h, they are first linearly projected to a set of small-patch conditioning vectors $c_b^{x_p, y_p}$, one for each small 2×2 patch (indexed by (x_p, y_p)) contained in the big 16×16 backbone patch. These are then simply added with the small-patch embeddings of $\mathbf{x}_t$ before feeding them to the convolutional stack. While this simple approach appears to work well, the alternative is to pass each small-patch vector

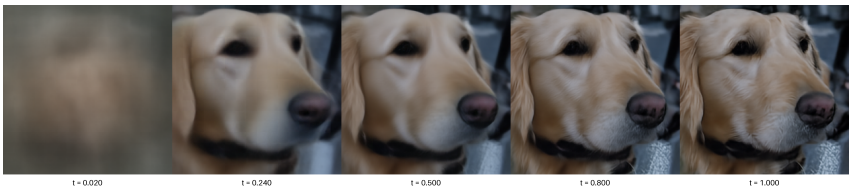


Figure 5: The predictions of the pretrained JiT-B/16 model at different timesteps t. Notice the smoothness of predictions at $t \leq 0.5$.

$c_b^{x_p, y_p}$ as conditioning to each convolutional block in the stack, where it is injected using AdaLN modulation. We tested this as well but found it to perform similarly and thus opted for the simpler additive version.

2.3 Late-stage pixel pathway

JiT, as well as other recent work (Yu et al., 2026; Chen et al., 2026c;b) enable the pixel pathway throughout the entire generation process. However, the additional pixel head can introduce substantial computational overhead. This begs the question: **Is the pixel head necessary at all denoising steps?** Since the pixel head is envisioned primarily as a component that generates the finer details in the image, we first look when the details emerge during the generative process and what predictions look like at different timesteps.

At early stages of denoising, the low signal-to-noise ratio of $\mathbf{x}_t$ leaves substantial uncertainty about the underlying clean image $\mathbf{x}$. The optimal solution for the model is then to predict the average over all clean images that can plausibly be reached from the given $\mathbf{x}_t$:

$$\arg\min_{\hat{\mathbf{x}}_\theta} \mathbb{E}\big[\|\hat{\mathbf{x}}_\theta - \mathbf{x}\|_2^2 \mid \mathbf{x}_t, t, c\big] = \mathbb{E}[\mathbf{x} \mid \mathbf{x}_t, t, c] \quad . \tag{3}$$

Since multiple clean images are compatible with the noisy $\mathbf{x}_t$, this averaging suppresses details, yielding smooth and blurry predictions, which is illustrated in Fig. 5. As denoising progresses and the signal-to-noise ratio increases, the uncertainty about these details decreases, allowing increasingly fine structure to emerge in the predicted clean images.

Late-Stage Pixel Head. Inspired by the above observations, we develop a novel approach that trains two different pixel heads: (1) a lightweight patch-overlapping linear head $g_{\text{linear}}(h, c)$, and (2) a convolutional head $g_{\text{conv}}(h, \mathbf{x}_t, t, c)$ as described above. During sampling, the linear head is active in earlier low-SNR denoising stages while the more expensive convolutional head is active in later high-SNR stages:

$$\hat{\mathbf{x}} = \begin{cases} \hat{\mathbf{x}}_{\text{lin}}, & t \leq t_a, \\ (1 - s(t))\hat{\mathbf{x}}_{\text{lin}} + s(t)\hat{\mathbf{x}}_{\text{conv}}, & t_a < t < t_b, \\ \hat{\mathbf{x}}_{\text{conv}}, & t \geq t_b, \end{cases} \tag{4}$$

where $\hat{\mathbf{x}}_{\text{lin}} = g_{\text{lin}}(h, c)$ and $\hat{\mathbf{x}}_{\text{conv}} = g_{\text{conv}}(h, \mathbf{x}_t, c, t)$ and $s(t)$ is a smooth interpolation function in the overlapping region. Note that we train the convolutional head over the full timestep range and set the linear head's loss weight to zero for $t \geq 0.8$.

The **overlapping linear output layer** g_{linear} used in the earlier stages is simply a linear layer, using the same implementation as JiT. However, instead of projecting backbone features h to $\mathbb{R}^{16\times16\times3}$ that specifies values for each backbone patch, it also generates values for K pixels surrounding the original patch. So, $g_{\text{linear}} : \mathbb{R}^{d_{\text{transformer}}} \to \mathbb{R}^{(16+2K)\times(16+2K)\times3}$. The final image is composed by smoothly interpolating between the different patches in the overlapping regions. While the additional overhead of predicting overlapping regions with other patches is negligible, it helps smoothen predictions and prevent patch boundary discontinuities in early denoising steps.

Applicability of Late-Stage Pixel Pathway. Note that this dual-head approach is only feasible when training with the x-prediction objective. When using v- or ϵ-prediction, the model is expected

(a) Quantitative comparison

Model	τ_{clip}	σ_m	FID ↓	IS ↑
JiT-B/16	0.05	0	**3.64**	**269.0**
	0	0	5.43	251.8
	0	0.018	3.81	263.1
JiT-L/16	0.05	0	**2.39**	**290.6**
	0	0	4.45	266.4
	0	0.018	2.72	281.7

(b–d) Qualitative comparison



(b) Clamp **(c)** No clamp **(d)** No clamp + Noise

Figure 6: Effect of denominator clamping during 50-step Heun sampling. **(a)** Quantitative results for pretrained JiT models with the original clamp ($\tau_{\text{clip}} = 0.05$), unclamped sampling, and matched-noise controls. CFG and EMA settings follow Li & He (2026). **(b–d)** JiT-L/16 samples with original clamp, no clamp, and no clamp followed by Gaussian noise with $\sigma_m = 0.018$, respectively.

to reproduce the exact original noise component present in the prediction target. However, as Li & He (2026) have shown, this is infeasible with a large-patch transformer with a linear output layer.

3 Denoising correction

Before discussing the training method, we first take a detour to JiT's (Li & He, 2026) sampling procedure. We observed that, in the original JiT codebase, the denominator $1 - t$ used to recover the velocity from predicted x is lower-bounded by $\tau_{\text{clip}} = 0.05$ during *both* training and sampling. While this stabilizes training by limiting the scaling of losses near $t = 1$, clipping during sampling modifies the sampling trajectory.

Normally, the final Euler step completes the denoising process and predicts the final image $\mathbf{x}_1 = \mathbf{x}_t + (1 - t)\hat{\mathbf{v}}_\theta = \hat{\mathbf{x}}_\theta$.

With the clipped denominator, however, the same update becomes

$$\mathbf{x}_1 = \mathbf{x}_t + \frac{1-t}{\max(1-t, \tau_{\text{clip}})}\left(\hat{\mathbf{x}}_\theta - \mathbf{x}_t\right) = \left(1 - \frac{1-t}{\max(1-t, \tau_{\text{clip}})}\right)\mathbf{x}_t + \frac{1-t}{\max(1-t, \tau_{\text{clip}})}\hat{\mathbf{x}}_\theta \ . \tag{5}$$

Whenever $1 - t < \tau_{\text{clip}}$, the sampler thus moves only a fraction $(1 - t)/\tau_{\text{clip}}$ toward the clean-image prediction, retaining part of the pre-terminal state. This is equivalent to terminating at a nonzero noise level because the clipping preserves part of the noise still present in $\mathbf{x}_t$. For example, with the default settings (50 steps, $\tau_{\text{clip}} = 0.05$), this gives $x_1 = 0.6 * x_{0.98} + 0.4\hat{\mathbf{x}}_\theta$. Note that the previous denoising step (from $x_{0.96}$) is also affected by clipping.

Does clamping $(\mathbf{1 - t})$ in the velocity conversion during sampling improve FID and why? We compare the following three variants of the sampling procedure: (1) the original clamped version ($\tau_{\text{clip}} = 0.05$), (2) the unclamped version ($\tau_{\text{clip}} = 0.0$), and (3) a version where we perform unclamped generation but subsequently add independent Gaussian noise to match the noise level of the original version.

The results are shown for the pre-trained JiT-B/16 and L/16 models in Fig. 6. For JiT-B/16, removing the clamp increases FID from 3.63 to 5.56 (keeping the same CFG), confirming that the original sampling modification has a substantial effect on the measured image quality. Interestingly, adding independent Gaussian noise to the fully denoised samples reduces FID to 3.84, recovering approximately 89% of the FID degradation caused by removing the clamp. This supports the hypothesis that the original clamped sampler benefits from preserving some random noise, rather than from producing a more reliable x_1 estimate.

Denoising correction. In our experiments, unless stated otherwise, we *do not* clamp the denominator $(1 - t)$ when computing $\hat{\mathbf{v}}_\theta$ during sampling. The sampler then follows the same trajectory as the equivalent v-prediction model and the final prediction has no residual noise. Note that clamping is not needed for stability at sampling time. An Euler step from t to t', $\mathbf{x}_{t'} = \mathbf{x}_t + (t' - t)\,\hat{\mathbf{v}}_\theta = \mathbf{x}_t + \frac{t'-t}{1-t}\left(\hat{\mathbf{x}}_\theta - \mathbf{x}_t\right)$, already scales $\hat{\mathbf{v}}_\theta$ with $t' - t$, such that $\frac{t'-t}{1-t} \leq 1$.

4 FROM STRUCTURE LEARNING TO ENDPOINT REFINEMENT

We use **two-stage training**. This enables more efficient training by enabling the pixel pathway only in the second stage. In the **first stage**, the transformer backbone is pre-trained using JiT's logit-normal timestep sampling ($t \sim \mathrm{LogitNormal}(\text{-}0.8, 0.8^2)$)[1]. Note that JiT's timestep sampler assigns extremely little probability mass do to the final part of the denoising trajectory ($\Pr(t > 0.95) \approx 1.4 \times 10^{-6}$, about 35,000 times less likely than a uniform sampler), even though this region is where the model performs pixel-level refinement. Therefore, in the **second stage**, where we train the pixel head and fine-tune the transformer backbone, uniform timestep sampling ($t \sim \mathcal{U}(0, 1)$ with a Soft-Min-SNR loss weighting is used:

$$\mathcal{L}_{\mathbf{v}} = \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\, t\sim\mathcal{U}(0,1)} \left[w_{\mathrm{soft}}(t) \left\| \hat{\mathbf{x}}_\theta - \mathbf{x} \right\|_2^2 \right], \qquad w_{\mathrm{soft}}(t) = \frac{\rho(t)}{\rho(t) + \gamma}, \qquad \rho(t) = \frac{t^2}{(1-t)^2}. \quad (6)$$

Here, $\rho(t)$ denotes the signal-to-noise ratio at timestep t. In essence, this preserves a weight close to 1 near the clean endpoint ($t \to 1$) and downweighs the losses near the noisy endpoint ($t \to 0$). This is similar to the v-loss weighting but is bounded near $t = 1$ and does not require clipping, while still preventing noisy gradients near $t \to 0$ from dominating the updates. See Sec. B in the Appendix for more details.

5 RELATED WORK

Diffusion and score-based models generate images by reversing corruption (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021), while flow matching and rectified flow learn noise-to-data vector fields (Lipman et al., 2023; Liu et al., 2023). Latent diffusion (Rombach et al., 2022) and DiT (Peebles & Xie, 2023) enable efficient synthesis in autoencoder representations. **Pixel-space generation** avoids this compression: early examples include DDPM, ADM (Dhariwal & Nichol, 2021), and cascaded systems such as Imagen (Saharia et al., 2022) and DeepFloyd IF (DeepFloyd, 2023). Simple Diffusion (Hoogeboom et al., 2023), SiD2 (Hoogeboom et al., 2025), and HDiT (Crowson et al., 2024) are based on efficient high-resolution architectures and training recipes; Edify (Atzmon et al., 2024) introduces frequency-dependent Laplacian diffusion.

Several works have investigated **architectural choices for pixel diffusion models**. JiT (Li & He, 2026) enables plain large-patch Transformers through clean-image prediction. Its linear output projection reconstructs non-overlapping patches from individual backbone tokens, leaving boundary consistency and fine detail to patch-level representations. Alternative detail mechanisms include PixNerd's patch-wise neural fields (Wang et al., 2026), DiP's patch detailer (Chen et al., 2026c), and PixelDiT's pixel pathway (Yu et al., 2026). The latter communicates across patches by compacting pixel features into patch tokens, applying global attention, and expanding back to pixels. HyperDiT (He et al., 2026) connects fine-grained tokens to multilevel semantic features through cross-attention, while PixelU (Guo et al., 2026) combines x-prediction with a U-shaped backbone and shallow-to-deep skips. Our shallow convolutions instead directly mix neighboring spatial features across JiT's original patch boundaries.

Some recent work also investigate **improved training and post-training** for pixel diffusion models. DeCo (Ma et al., 2026b) separates semantic and high-frequency modeling; PixelGen (Ma et al., 2026c) adds perceptual supervision, and PixelREPA (Shin et al., 2026) adapts representation alignment to JiT using transformed targets and a masked adapter. Pixel-space supervision also improves post-training of latent models (Zhang et al., 2024), whereas L2P (Chen et al., 2026b) and Z-Image-Pixel (Jiang et al., 2026) investigate conversion of latent models to pixel-space generation. We instead fine-tune an already trained pixel-space JiT with an added refinement pathway, preserving the pretrained backbone as the starting point for learning local corrections.

Concerning **coarse-to-fine denoising**, PixelFlow (Chen et al., 2025), Scale Space Diffusion (Mukhopadhyay et al., 2026), and Spectral Progressive Diffusion (Xiao et al., 2026) reduce early computation through progressive resolution. Spectral Forcing (Fan et al., 2026) increases the input frequency bandwidth with denoising time, while Latent Forcing (Baade et al., 2026) coordinates latent and pixel noise schedules to establish structure before pixel detail. PixelGen (Ma et al.,

[1] In practice, we use the pre-trained checkpoints provided by Li & He (2026).

Table 1: Comparison of pixel-head variants for JiT-B and JiT-L. For Dual Head variants, GFLOPs are reported with the linear head and the convolutional head separately. Further details and breakdowns are provided in Sec. D.

Method	Params. (M)	GFLOPs (per FE)	Time ↓ (s/img)	gFID ↓	sFID ↓	IS ↑
JiT-B						
Pre-trained ($\epsilon = 0.05$)	131.3	50.4	1.44	3.64	7.21	269.0
Pre-trained ($\epsilon = 0.0$)	131.3	50.4	1.44	4.61	8.68	282.4
Fine-tuned (100 epochs)	131.3	50.4	1.44	4.29	7.21	275.7
Conv. pixel head	132.9	60.9	1.74	3.42	5.49	264.3
Dual pixel head ($0.5 - 0.7$)	135.4	50.8 / 60.9	1.55	3.41	5.71	269.3
Dual pixel head ($t_{\text{thresh}} = 0.8$)	135.4	50.8 / 60.9	1.49	3.42	6.38	270.7
JiT-L						
Pre-trained ($\epsilon = 0.05$)	459.1	175.9	3.08	2.39	6.28	290.6
Pre-trained ($\epsilon = 0.0$)	459.1	175.9	3.08	3.51	9.15	324.6
Fine-tuned (100 epochs)	459.1	175.9	3.08	2.72	6.87	309.3
Conv. pixel head	460.6	186.8	3.27	2.27	5.35	308.2
Dual pixel head ($0.5 - 0.7$)	464.5	176.4 / 186.8	3.21	2.24	5.44	309.8
Dual pixel head ($t_{\text{thresh}} = 0.8$)	464.5	176.4 / 186.8	3.16	2.20	5.98	307.5

2026c) restricts perceptual supervision to lower-noise timesteps. These methods motivate timestep-dependent processing but use different mechanisms from our head switch: at fixed image resolution, an overlapping linear head handles early denoising, followed by a convolutional pixel head in later steps. We study whether this late-only refinement retains the quality benefit of applying the added head throughout sampling.

6 Experiments

6.1 Experimental Setup

Training setup. For each experiment, we initialize the model from the pretrained JiT checkpoint and augment it with the randomly initialized layers introduced in Sec. 2. In the first stage of fine-tuning (15 epochs unless otherwise specified), only the added pixel pathway is trained while the backbone remains frozen. In the second stage of fine-tuning (85 epochs), the pretrained backbone is unfrozen and trained together with the pixel pathway. The learning rate is 5×10^{-5} for the pretrained backbone and 1×10^{-4} for the newly introduced layers. The global batch size is fixed at 256, while all remaining training hyperparameters follow the defaults of the original JiT implementation. We do not use any additional losses, such as the representation alignment loss (REPA (Yu et al., 2025)).

Evaluation setup. For trained models, we run a sweep on EMA and CFG settings. First, we sweep different combinations of the varied hyperparameters with 10k images generated for each setting. The best setting is selected and 50k images are generated, which are used to measure generator quality using FID and IS like previous work. Generally, we sweep CFG's in the range 2.0 to 3.5 and try it with both EMA weights (decays of 0.9996 and 0.9999). During generation, we use JiT's default Heun (Karras et al., 2022) sampler with 50 steps.

6.2 Results

Tab. 1 shows the effect of introducing the pixel pathway and its late application. Qualitative results are shown in Appendix D.4.

Convolutional pixel pathway. Comparing the metrics for the pre-trained model with the "Conv. pixel head" rows for both JiT-B and JiT-L in Tab. 1 reveals that there is indeed improvement in image quality, as measured by FID. In particular, sFID, which is more attuned to superficial image

Table 2: Comparison on ImageNet@256 for recent pixel-space generative models. For full comparison, see Tab. 5 in Sec. A. For the H-model, we use AdaLN-based conditioning and wider pixel head (BW-Cond variant; see Sec. C) and $t \sim \mathrm{LogitNormal}(0, 1)$. The convolutional head is activated at $t = 0.5$.

Method		REPA	Epochs	Params	NFE	gFID↓	IS↑
Simple Diffusion	(Hoogeboom et al., 2023)	–	800	2.0B	–	2.44	256.3
PixelDiT-XL	(Yu et al., 2026)	✓	320	797M	100×2	1.61	292.7
DeCo-XL/16	(Ma et al., 2026b)	✓	600	682M	100×2	1.69	304.0
DiP-XL/16	(Chen et al., 2026c)	✓	600	631M	100×2	1.79	281.9
PixelU-H/16	(Guo et al., 2026)	✓	600	1.17B	100×2	1.63	305.9
JiT-H/16	(Li & He, 2026)	–	600	953M	100×2	1.86	303.4
JiT-H/16 ($\epsilon = 0.0$)	Reproduction	–	600	953M	100×2	2.34	335.4
JiT-H/16 + Dual Head	Ours	–	600+100	976M	100×2	1.78	333.6

quality, shows a larger drop. Interestingly, the post-trained original JiT model control is also further improved in post-training, but still doesn't match the original numbers. In fact, the gFID pushes below the originally reported results from Li & He (2026). The parameter count increase is negligible compared to total parameter count. Generation runtime increases 21% for JiT-B and 6.2% for JiT-L. Note that for both backbones, we used a small-sized pixel pathway with only 4 dense convolutions.

Late pixel pathway. The results for the dual pixel head rows reveal that there is little to no quality degradation when the convolutional pixel pathway is applied only in the last 20% of the denoising trajectory. Introducing the late pixel pathway with the second overlapping linear output head further increases the parameter count but results in a reduction in generation speed for both models. Even though the difference is small for the L model since the small convolutional head is inexpensive compared to the total runtime of JiT-L, this difference can grow much larger if a more expensive pixel head is used. For example, an architectural estimate for PixelDiT-XL gives 220 GFLOPs for their backbone and 90 GFLOPs for their pixel pathway. This means that the pixel pathway consumes nearly 30% of total compute. With the late pixel pathway, this can be reduced to just around 8% of total compute with the same head, which gives 23% total savings in GFLOPs.

6.3 Comparison

A comparison to previous work is shown in Tab. 2. A full comparison is provided in Tab. 5 in Sec. A in the Appendix. JiT-H/16 obtains an FID of 2.34 with the corrected (unclamped) sampler. Our method (JiT-H/16 + Dual Head, starting at $t = 0.5$) obtains an FID of 1.78, beating the original JiT-H/16 and JiT-G/16 models. Note that most other previous works used REPA losses to further improve their results. Nevertheless, our numbers remain competitive and illustrate our main finding of improving a pretrained JiT checkpoint with a late pixel pathway extension.

6.4 Ablations

Starting point of the late pixel pathway. We vary the inference-time transition while keeping the trained JiT-B checkpoint fixed (Tab. 3). Restricting the pixel pathway to the final 20% of the trajectory with a hard switch at $t = 0.8$ yields a gFID of 3.42, compared to 3.39 for a smooth transition over $[0.5, 0.6]$. However, delaying the transition to $[0.9, 0.99]$ increases gFID to 5.39, suggesting that the pixel pathway requires sufficient time for late-stage refinement.

No skip from $\mathbf{x}_t$. In Eq. (2), the pixel head receives the noisy pixel input $\mathbf{x}_t$ directly. This shortcut essentially bypasses the original big-patch backbone while remaining conditioned on the large-patch backbone's features. Related work (DiP, PixelDiT) discusses the benefit of explicit local modeling under a v-prediction approach but does not study whether direct access to $\mathbf{x}_t$ remains equally important under JiT's x-prediction parameterization. Under JiT's manifold intuition, x-prediction avoids directly predicting the exact noise component present in the velocity target, potentially reducing the need for a high-frequency bypass.

Table 3: Transition ablation for JiT-B and JiT-L with Dual Pixel Heads on ImageNet 256×256, measured using gFID with 50k samples. Smooth transitions blend the linear and pixel heads over the indicated interval; hard switches activate the pixel head at the indicated timestep.

	Gradual					Hard				
Method	.50–.60	.60–.70	.70–.80	.80–.90	.90–.99	.60	.70	.80	.90	.95
JiT-B + Dual Pixel Head	3.39	3.41	3.41	3.46	5.39	3.40	3.41	3.42	3.58	4.98
JiT-L + Dual Pixel Head	2.24	2.23	2.21	2.20	3.55	2.24	2.22	2.20	2.27	3.24

Table 4: Ablation results on $\mathbf{x}_t$ skip connection (left) and residual pathway (right).

JiT-B + Dual Pixel Head	gFID ↓	IS ↑
Baseline	3.41	269.3
No skip from $\mathbf{x}_t$	3.91 (+0.50)	277.9

JiT-B + Conv Head (Residual)	gFID ↓	IS ↑
Baseline	3.53	266.5
Backbone frozen	4.20 (+0.67)	283.6

The second row in Tab. 4 shows the effect of not feeding $\mathbf{x}_t$ directly to the pixel head. The FID degrades from 3.41 to 3.91. However, this is not as detrimental as disabling $(1 - t)$ clipping (FID 4.61), suggesting that some image quality can be recovered through architecture choices alone. While we couldn't find similar ablations for v-prediction objectives in previous work, we expect that the FID drop in those cases will be more significant since $\mathbf{x}_t$ is important for predicting the exact noise pattern still present in $\mathbf{x}_t$.

Residual pixel pathway. We also experiment with a residual formulation for the pixel head, where it adds to the original prediction. Third row of Tab. 4 shows that this also achieves good image quality and is thus a viable alternative to simply replacing the original output layer. The fourth row reports results for using the residual formulation but keeping the backbone frozen. The relatively poor result (FID 4.20) indicates that original predictions can not be easily corrected with a simple post-hoc addition and requires backbone fine-tuning.

Additional ablations. We present additional ablations and analysis in the Appendix. Sec. C presents ablations on the pixel head design, including its size. Sec. B presents an analysis on the effect of timestep sampling distributions and loss weights in the fine-tuning stage.

7 Limitations

Our study prioritizes controlled comparisons with the pretrained JiT baselines in order to improve the understanding of pixel pathways. Our best results can likely be further improved by using more advanced training techniques and higher-capacity pixel heads. Some open questions remain. We do not establish the optimal point during training at which to introduce the pixel pathway during training. Our exploration of alternative pixel head designs is limited, leaving the potential benefits of higher-capacity designs unresolved. Scaling beyond the evaluated model sizes and to text-to-image models and adapting pretrained latent-space diffusion models are other directions for future work.

8 Conclusion

In this work, we showed that fine-tuning a simple convolutional pixel head on top of pretrained JiT models can significantly improve image quality. In addition, we find that the pixel pathway is not necessary throughout the entire denoising process, and can be activated as late as in the last 20% of the denoising trajectory, which reduces computation requirements and further illustrates the usefulness of the x-prediction objective for pixel-space modeling. We hope our findings inspire future work in this area and lead to further improvements in pixel-space diffusion transformers.

AI USE STATEMENT

AI-based tools were used during the preparation of this work to assist with writing, in particular the formatting and improvement of result tables, diagrams, drafting/rephrasing some paragraphs and reviewing written text. All generated tables, diagrams and text were manually reviewed and adjusted or reformulated to ensure correctness and improve clarity of presentation. The core research ideas and approach were developed independently, with AI-assisted search being used to verify and find certain details in literature. Experimental settings were determined based on previous work. All experiments have been designed, executed and interpreted manually by the authors. The authors assume full responsibility for the final manuscript.

AI-based tools were also used to write several ablation and analysis scripts (e.g. convolutional head transition sweep and progression of x_0 predictions) and modify previous manually written code for additional features. Core parts of the code (training, model, inference) are based on code previously provided by Li & He (2026), and were modified mostly manually, except for certain secondary features (e.g. reproducible seeds per image). AI assisted during manual code review to double-check correctness and identify possible implementation errors. All AI-generated code was manually reviewed by the main author.

ETHICS STATEMENT

This work does not involve human participants, private or sensitive information, or experiments that present foreseeable risks to individuals. Our evaluation is based entirely on publicly available datasets and pretrained models, used under their applicable licenses and usage conditions. We follow established research practices in the design, evaluation, and reporting of our experiments, with an emphasis on accurate and transparent presentation of the results. We have not identified any ethical concerns specific to the proposed approach beyond the broader considerations inherent to the datasets and models on which it builds.

REPRODUCIBILITY STATEMENT

Our experiments use ImageNet and publicly available pretrained JiT checkpoints. We describe the architectural modifications in Section 2, the sampling correction in Section 3, and the training objective in Section 4. Section 6.1 specifies the fine-tuning schedule, learning rates, batch size, and evaluation protocol, including EMA and classifier-free guidance sweeps. Computational cost breakdowns are provided in Appendix D. We will release our code and model checkpoints.
Anonymized code is provided at `https://anonymous.4open.science/r/JiT-BD8B`.

AUTHOR CONTRIBUTIONS

Anonymized for review.

ACKNOWLEDGMENTS

Anonymized for review.

REFERENCES

Yuval Atzmon, Maciej Bala, Yogesh Balaji, Tiffany Cai, Yin Cui, Jiaojiao Fan, Yunhao Ge, Siddharth Gururani, Jacob Huffman, Ronald Isaac, Pooya Jannaty, Tero Karras, Grace Lam, J. P. Lewis, Aaron Licata, Yen-Chen Lin, Ming-Yu Liu, Qianli Ma, Arun Mallya, Ashlee Martino-Tarr, Doug Mendez, Seungjun Nah, Chris Pruett, Fitsum Reda, Jiaming Song, Ting-Chun Wang, Fangyin Wei, Xiaohui Zeng, Yu Zeng, and Qinsheng Zhang. Edify Image: High-Quality Image Generation with Pixel Space Laplacian Diffusion Models. *arXiv:2411.07126*, 2024.

Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Li Fei-Fei. Latent Forcing: Reordering the Diffusion Trajectory for Pixel-Space Image Generation. In *International Conference on Machine Learning (ICML)*, 2026.

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

Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. PixelFlow: Pixel-Space Generative Models with Flow. *arXiv:2504.07963*, 2025.

Zhennan Chen, Junwei Zhu, Xu Chen, Jiangning Zhang, Jiawei Chen, Zhuoqi Zeng, Wei Zhang, Chengjie Wang, Jian Yang, and Ying Tai. L2P: Unlocking Latent Potential for Pixel Generation. *arXiv:2605.12013*, 2026b.

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 (CVPR)*, 2026c.

Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z. Kaplan, and Enrico Shippole. Scalable High-Resolution Pixel-Space Image Synthesis with Hourglass Diffusion Transformers. In *International Conference on Machine Learning (ICML)*, volume 235, 2024.

DeepFloyd. DeepFloyd IF. Official model repository, 2023. URL `https://github.com/deep-floyd/IF`.

Prafulla Dhariwal and Alexander Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Weichen Fan, Haiwen Diao, Penghao Wu, and Ziwei Liu. Show the Signal, Hide the Noise: Spectral Forcing for Pixel-Space Diffusion. *arXiv preprint arXiv:2606.15236*, 2026.

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. In *European Conference on Computer Vision (ECCV)*, 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:2605.15741*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning (ICML)*, 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 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Dengyang Jiang, Ruoyi Du, Zhennan Chen, Dongyang Liu, Zanyi Wang, Mingzhe Zheng, Xiangpeng Yang, Huanqia Cai, Aiming Hao, Yuming Jiang, Peng Gao, Harry Yang, and Steven Hoi. An Empirical Study of Training Pixel-Space Text-to-Image Diffusion Models. *arXiv:2608.16887*, 2026.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the Design Space of Diffusion-Based Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

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 (CVPR)*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Lichen Ma, Zipeng Guo, Yu He, Xiaolong Fu, Luohang Liu, Jingling Fu, Junshi Huang, and Yan Li. FrequencyBooster: Full-Frequency Modeling for High-Fidelity Pixel Diffusion. *arXiv:2605.17759*, 2026a.

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 (ECCV)*, 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 (CVPR)*, 2026b.

Zehong Ma, Ruihan Xu, and Shiliang Zhang. PixelGen: Improving Pixel Diffusion with Perceptual Supervision. *arXiv:2602.02493*, 2026c.

Soumik Mukhopadhyay, Prateksha Udhayanan, and Abhinav Shrivastava. Scale Space Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

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 (CVPR)*, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Raphael Gontijo-Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Axel Sauer, Katja Schwarz, and Andreas Geiger. StyleGAN-XL: Scaling StyleGAN to Large Diverse Datasets. In *ACM SIGGRAPH Conference Proceedings*, 2022.

Jaeyo Shin, Jiwook Kim, and Hyunjung Shim. Representation Alignment for Just Image Transformers is not Easier Than You Think. In *European Conference on Computer Vision (ECCV)*, 2026.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. PixNerd: Pixel Neural Field Diffusion. In *International Conference on Learning Representations (ICLR)*, 2026.

Howard Xiao, Brian Chao, Lior Yariv, and Gordon Wetzstein. Spectral Progressive Diffusion for Efficient Image and Video Generation. *arXiv:2605.18736*, 2026.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. Generation: Taming Optimization Dilemma in Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Weiyi You, Jinhua Zhang, Xingyu Zhou, Wei Long, Junyu Lou, and Shuhang Gu. Advanced Pixel Diffusion Model with Guided Sparse Global Refinement. *arXiv:2609.00798*, 2026.

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 *International Conference on Learning Representations (ICLR)*, 2025.

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 (CVPR)*, 2026.

Christina Zhang, Simran Motwani, Matthew Yu, Ji Hou, Felix Juefei-Xu, Sam Tsai, Peter Vajda, Zijian He, and Jialiang Wang. Pixel-Space Post-Training of Latent Diffusion Models. *arXiv:2409.17565*, 2024.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion Transformers with Representation Autoencoders. In *International Conference on Learning Representations (ICLR)*, 2026.

Hongkai Zheng, Weili Nie, Arash Vahdat, and Anima Anandkumar. Fast Training of Diffusion Models with Masked Transformers. *Transactions on Machine Learning Research (TMLR)*, 2024.

A Full Comparison

Table 5: Full comparison on 256×256 generation on ImageNet. NFE values with $\times 2$ denote nominal counts with classifier-free guidance; guidance intervals may reduce the actual count. RAE uses AutoGuidance with an additional guiding model.

Method		Epochs	Params	NFE	gFID↓	IS↑
Latent Generative Models						
DiT-XL	(Peebles & Xie, 2023)	1400	675M + 49M	250×2	2.27	278.2
SiT-XL	(Ma et al., 2024)	1400	675M + 49M	250×2	2.06	277.5
MaskDiT	(Zheng et al., 2024)	1600	675M + 49M	79×2	2.28	276.6
REPA	(Yu et al., 2025)	800	675M + 49M	250×2	1.42	305.7
LightningDiT	(Yao et al., 2025)	800	675M + 41M	250×2	1.35	295.3
RAE-XL	(Zheng et al., 2026)	800	839M + 415M	–	**1.13**	262.6
Pixel Generative Models						
StyleGAN-XL	(Sauer et al., 2022)	–	–	–	2.30	265.1
Simple Diffusion	(Hoogeboom et al., 2023)	800	2.0B	–	2.44	256.3
PixelFlow-XL	(Chen et al., 2025)	320	677M	Adaptive	1.98	282.1
PixNerd-XL	(Wang et al., 2026)	320	700M	100×2	1.93	298.0
PixelDiT-XL	(Yu et al., 2026)	320	797M	100×2	1.61	292.7
PixelGen-XL/16	(Ma et al., 2026c)	160	676M	100×2	1.83	293.6
DeCo-XL/16	(Ma et al., 2026b)	600	682M	100×2	1.69	304.0
DiP-XL/16	(Chen et al., 2026c)	600	631M	100×2	1.79	281.9
PixelREPA-H/16	(Shin et al., 2026)	600	953M	100×2	1.81	317.2
AsymFlow-H/16	(Chen et al., 2026a)	600	953M	100×2	1.57	–
FrequencyBooster-H/16	(Ma et al., 2026a)	320	1.1B	100×2	1.60	321.0
PixelU-H/16	(Guo et al., 2026)	600	1.17B	100×2	1.63	305.9
PixSGR-H/16	(You et al., 2026)	320	875M	100×2	**1.51**	311.0
JiT Models						
JiT-H/16	(Li & He, 2026)	600	953M	100×2	1.86	303.4
JiT-G/16	(Li & He, 2026)	600	2B	100×2	1.82	292.6
JiT-H/16 ($\epsilon = 0.0$)	Reproduction	600	953M	100×2	2.34	335.4
JiT-H/16 ($\epsilon = 0.05$)	Reproduction	600	953M	100×2	1.88	302.7
JiT-H/16 + Dual Head	Ours	600+100	976M	100×2	**1.78**	333.6

A full comparison is provided in Tab. 5.

B Timestep sampling and loss weighting

Sparse endpoint coverage. JiT samples training timesteps t from a logit-normal distribution, $t \sim \mathrm{LogitNormal}(\text{-}0.8, 0.8^2)$. This distribution assigns extremely little probability to the final part of the denoising path:

$$\Pr(t > 0.9) \approx 9.0 \times 10^{-5}, \qquad \Pr(t > 0.95) \approx 1.4 \times 10^{-6}. \tag{7}$$

This means that JiT's logit-normal sampler visits $t > 0.95$ approximately $35{,}000$ times less frequently than a uniform sampler. Although these intervals constitute only a small portion of the flow, they correspond to low-noise states close to the data endpoint, where the model must perform accurate late-stage refinement. Undersampling in this region can be especially consequential in pixel-space generative models where residual prediction errors directly affect fine image details.

Other works (Yu et al., 2026; Chen et al., 2026c) are better in this regard since they use logit-normal timestep sampling with $\mathcal{N}(0, 1)$, which doesn't suppress the near-clean region ($t > 0.9$) as strongly.

Stronger endpoint coverage can be achieved by using using uniformly sampled timesteps ($t \sim \mathcal{U}(0, 1)$).

To understand the effect of timestep sampling distribution and loss weighting, in this work, we experiment with these three options:

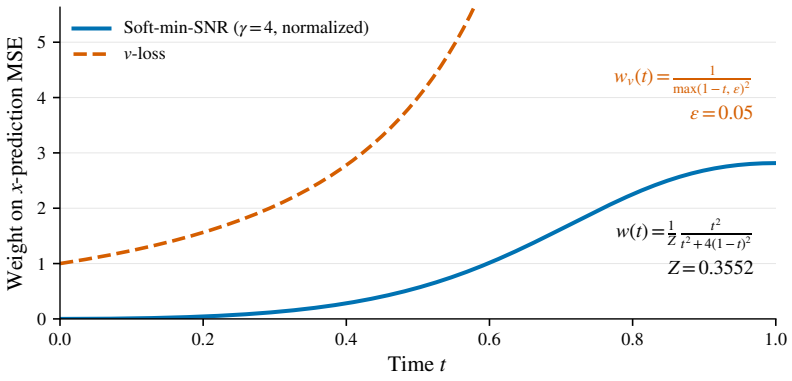


Figure 7: Weights applied to the x-prediction MSE: normalized soft-min-SNR ($\gamma = 4$) and v-loss. The v-loss curve is clipped at the plot boundary.

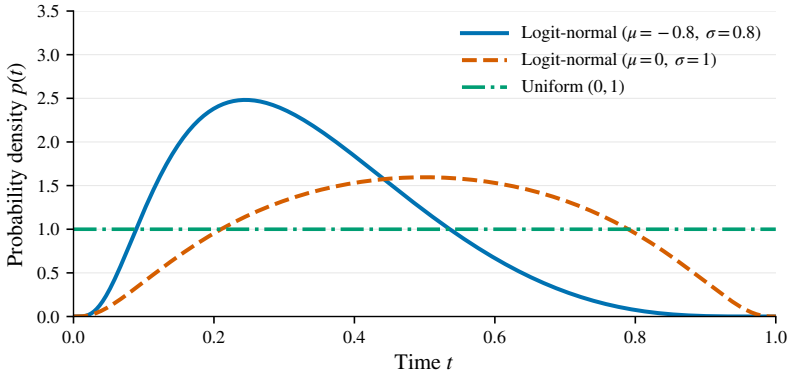


Figure 8: Timestep sampling densities for logit-normal distributions with $(\mu, \sigma) = (-0.8, 0.8)$ and $(0, 1)$, and uniform sampling on $[0, 1]$.

1. original $t \sim \text{LogitNormal}(\text{-}0.8, 0.8^2)$ with v-loss (Eq. (9)).
2. standard $t \sim \text{LogitNormal}(0, 1)$ with v-loss.
3. $t \sim \mathcal{U}(0, 1)$ with Soft-Min-SNR weighting

The timestep sampling distributions and the loss weights are plotted in Fig. 8 and Fig. 7, respectively.

Loss weighting. A model that predicts the clean endpoint x can be trained using a weighted MSE loss:

$$\mathcal{L} = w(t)\,\|\hat{\mathbf{x}}_\theta - \mathbf{x}\|_2^2\,. \tag{8}$$

When converting the v-loss MSE objective $\mathcal{L} = \mathbb{E}[\|\hat{v}_\theta - \mathbf{v}\|_2^2]$ to x-prediction, the following loss emerges:

$$\mathcal{L}_\mathbf{v} = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D},t\sim\mathcal{T}}\left[w(t)\,\|\hat{\mathbf{x}}_\theta - \mathbf{x}\|_2^2\right], \qquad w_{\text{v-loss}}(t) = \frac{1}{(1-t)^2}. \tag{9}$$

In practice, the denominator should be clipped to ensure numerical stability.

Soft-Min-SNR endpoint refinement. Uniform sampling alone does not ensure that endpoint errors receive sufficient optimization pressure. Direct clean-image prediction is more difficult at low t, where there is considerable noise in the image. This leaves the model with high uncertainty about prediction target and therefore the regression errors become large. On the other hand, because

of the low noise, errors near $t = 1$ are typically much smaller since the image has already been largely formed. With uniformly weighted clean-image objective, the errors near $t = 0$ may therefore dominate the updates, slowing convergence near $t = 1$. To remedy this, timestep-dependent error weighting can be applied that suppresses noisy regions near $t = 0$.

We combine uniform sampling with a Soft-Min-SNR-weighted loss weighting. For JiT's linear interpolation path, we define

$$\mathrm{SNR}(t) = \frac{t^2}{(1-t)^2} \tag{10}$$

and optimize

$$\mathcal{L}_{\mathrm{soft}} = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\left[\frac{w(t)}{Z}\,\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t, t) - \mathbf{x}\|_2^2\right], \tag{11}$$

where

$$w(t) = \frac{\mathrm{SNR}(t)}{\mathrm{SNR}(t)+\gamma}, \qquad Z = \mathbb{E}_{t\sim\mathcal{U}(0,1)}\left[w(t)\right]. \tag{12}$$

The normalization Z keeps the expected loss weight constant, preventing the change of objective from implicitly modifying the effective learning rate.

This weighting suppresses the large and noisy errors near $t = 0$ while increasing the relative contribution of errors near $t = 1$. Compared to the inverse-square weighting induced by JiT's $\mathbf{v}$-loss, it remains bounded as $t \to 1$ and doesn't require denominator clipping. The same functional form has appeared as Soft-Min-SNR weighting in pixel-space diffusion (Crowson et al., 2024).

B.1 Ablation

Table 6: Comparison of timestep sampling and loss weighting on ImageNet 256×256 using the dual-head JiT-B/16 model with the B head. We select the lowest FID from final-epoch CFG/EMA sweeps from JiT's implementation, evaluated on 50,000 images and report IS from the same evaluation. All selected results use 50-step Heun sampling and decoder-transition start 0.5 and end 0.7.

Timestep distribution	Loss weighting	FID ↓	IS ↑
Logit-normal ($\mu = -0.8, \sigma = 0.8$)	v-loss	3.65	282.86
Logit-normal ($\mu = 0, \sigma = 1$)	v-loss	3.41	274.04
Uniform*	v-loss	4.01	285.48
Uniform	Soft-min-SNR ($\gamma = 4$)	3.41	269.34

Tab. 6 shows a comparison of timestep sampling and loss weighting using the dual-head JiT-B/16 model with the B head.

C Additional Pixel Head Ablations

Tab. 7 shows several ablations on the pixel head.

Head Size. For head sizes for the main ConvNeXt variant, we experimented with T(iny), B(ase), B(ase)-W(ide) and L(arge). Each block contains only one dense 3×3 convolution, followed by two 1×1 convolutions. Depth in the table refers to the number of such blocks. From the results, we see that the B and BW variants provide best performance, with T being too small. ResNet-M uses two residual blocks, each with two 3×3 convolutions.

Conditioning. We also experimented with using backbone features to condition the convolutional blocks using AdaLN modulation. These are reported as "Cond" variants. From our exploration, they do not appear to improve performance.

ResNet vs ConvNeXt. Finally, we also experimented with using ResNet blocks, reported as ResNet. We matched the receptive fields by using two blocks, each containing two convolutions. From the results it appears that they do not perform better than our main ConvNeXt-like variant.

Table 7: Head configuration ablation on ImageNet 256×256. We report FID and Inception Score (IS) on 50,000 generated images using 50-step Heun sampling. For each configuration, we select the lowest-FID evaluation from CFG/EMA sweeps using JiT's evaluator, and report its corresponding IS. Width and depth specify the head channel width and number of blocks. Head parameters (millions) include all weights and biases in the convolutional or ResNet head, excluding the backbone and overlap head. Head GFLOPs count convolution and matrix-multiplication operations per image for one forward pass. Head share is relative to the full forward pass. All runs use uniform timestep sampling and soft-min-SNR loss ($\gamma = 4$).

Backbone	Head	Width	Depth	Head params (M)	Head GFLOPs	Share (%)	FID ↓	IS ↑
B/16	T	64	2	3.36	6.22	10.9	3.71	274.15
B/16	B	64	4	3.53	10.79	17.5	3.41	269.34
B/16	B-Cond	64	4	6.77	15.62	23.5	3.46	273.43
B/16	BW	128	4	7.71	39.83	43.9	3.28	272.28
B/16	BW-Cond	128	4	14.30	55.94	52.4	3.37	281.83
B/16	L	128	6	8.37	58.08	53.3	3.51	282.14
B/16	ResNet-M	96	2	5.22	13.36	20.8	3.69	270.17
L/16	B	64	4	4.59	11.32	6.0	2.24	309.83
L/16	B-Cond	64	4	8.90	16.69	8.6	2.18	308.18
L/16	ResNet-M	96	2	6.82	14.17	7.4	2.30	306.00

D Computational Cost Breakdown of ConvDual Models

We first describe the computational resources used in our experiments. Then we report the computational cost of ConvDual-B/16, ConvDual-L/16, and ConvDual-H/16 at an image resolution of 256×256.

D.1 Compute Resources

We conducted large-scale training on single nodes located in a computing cluster equipped with four NVIDIA H100 GPUs and two AMD EPYC 9454 (48 core) CPUs per node. Initial experimentation and debugging were performed on a server with two 32-core AMD EPYC 7452 CPUs and four NVIDIA A40 GPUs (48 GB of VRAM each), running Ubuntu 20.04.6 LTS.

D.2 Cost per network evaluation

Table 8: Computational cost per network forward pass. Values are GFLOPs; percentages indicate the fraction of the corresponding row's total. "Heads" includes all active output heads. One CFG-guided evaluation requires two network forwards and therefore doubles all listed costs.

Model	Active heads	Total	Backbone	Heads
B/16	Linear	50.82	50.14 (98.66%)	0.68 (1.34%)
	Conv	60.93	50.14 (82.29%)	10.79 (17.71%)
	Both	61.61	50.14 (81.38%)	11.47 (18.62%)
L/16	Linear	176.39	175.48 (99.48%)	0.91 (0.52%)
	Conv	186.81	175.48 (93.94%)	11.32 (6.06%)
	Both	187.72	175.48 (93.48%)	12.24 (6.52%)
H/16	Linear	364.52	363.38 (99.69%)	1.14 (0.31%)
	Conv	423.61	363.38 (85.78%)	60.23 (14.22%)
	Both	424.75	363.38 (85.55%)	61.37 (14.45%)

Table 8 reports the cost of one network forward pass, before classifier-free guidance (CFG). When both output heads are active, they share a single backbone evaluation.

The linear head costs only 0.682 GFLOPs for ConvDual-B and 0.910 GFLOPs for ConvDual-L, compared with 10.788 and 11.325 GFLOPs, respectively, for the convolutional head. Consequently, evaluating both heads is only slightly more expensive than evaluating the convolutional head alone.

D.3 COST PER GENERATED IMAGE.

Table 9: Estimated computation per generated image using 50 sampling steps with Heun, a final Euler step, and CFG. The convolutional head variant is shown below each model. Values are TFLOPs; percentages give each component's share of the total. Head costs aggregate all active head evaluations over the sampling trajectory.

Model	Head schedule	Total	Backbone	Heads
B/16	Conv throughout	12.1	9.9 (82.3%)	2.1 (17.7%)
Head: B	Blend 0.5–0.7	11.1	9.9 (89.8%)	1.1 (10.2%)
	Switch at 0.8	10.5	9.9 (94.8%)	0.5 (5.2%)
L/16	Conv throughout	37.0	34.7 (93.9%)	2.2 (6.1%)
Head: B	Blend 0.5–0.7	36.0	34.7 (96.6%)	1.2 (3.4%)
	Switch at 0.8	35.3	34.7 (98.3%)	0.6 (1.7%)
H/16	Conv throughout	83.9	71.9 (85.8%)	11.9 (14.2%)
Head: BW-Cond	Blend 0.5–0.7	77.9	71.9 (92.4%)	5.9 (7.6%)
	Switch at 0.8	74.4	71.9 (96.7%)	2.5 (3.3%)

We use the same sampling configuration as the other experiments: 50 denoising steps, with Heun updates for the first 49 steps and an Euler update for the final step. This requires 99 guided function evaluations. Each guided evaluation performs a conditional and an unconditional network forward, yielding 198 network forwards per generated image.

We compare three head schedules as sampling time increases from $t = 0$ to $t = 1$: (i) the convolutional head throughout; (ii) a transition from the linear overlapping head to the convolutional head over $t \in [0.5, 0.7]$; and (iii) a hard switch from linear to convolutional predictions at $t = 0.8$. For the transition schedule, only the linear head is used at $t \leq 0.5$, both heads are used at $0.5 < t < 0.7$, and only the convolutional head is used at $t \geq 0.7$. For the hard switch, the convolutional head is used at $t \geq 0.8$.

Relative to using the convolutional head throughout, the transition schedule reduces total computation by 8.34%, 2.78%, and 7.13% for ConvDual-B, ConvDual-L, and ConvDual-H, respectively. The hard switch at $t = 0.8$ increases these savings to 13.24%, 4.45%, and 11.27%, respectively. The backbone cost remains constant across schedules; all computational savings come from changing which output heads are evaluated.

D.4 QUALITATIVE EXAMPLES



class 12: house finch, linnet, Carpodacus mexicanus



class 14: indigo bunting, indigo finch, indigo bird, Passerina cyanea



class 42: agama



class 81: ptarmigan



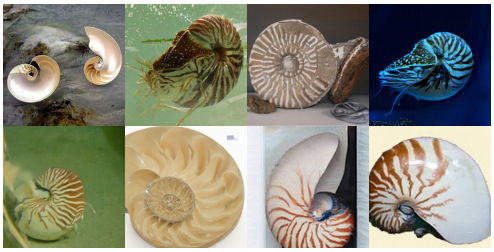
class 107: jellyfish



class 108: sea anemone, anemone



class 110: flatworm, platyhelminth



class 117: chambered nautilus, pearly nautilus, nautilus



class 130: flamingo

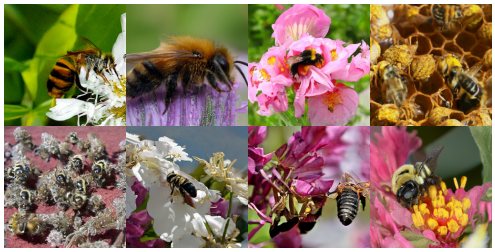


class 279: Arctic fox, white fox, Alopex lagopus

Figure 9: Qualitative samples on ImageNet 256×256, conditioned on the indicated classes, generated with JiT-H/16 + Dual Head and CFG of 2.8.



class 288: leopard, Panthera pardus



class 309: bee



class 349: bighorn, bighorn sheep, cimarron, Rocky Mountain bighorn, Rocky Mountain sheep, Ovis canadensis



class 397: puffer, pufferfish, blowfish, globefish



class 425: barn



class 448: birdhouse



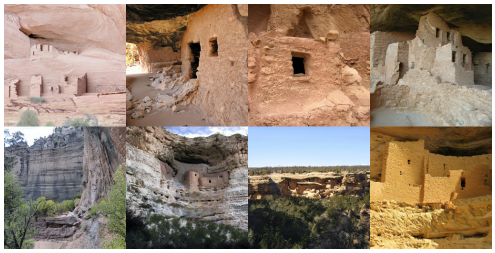
class 453: bookcase



class 458: brass, memorial tablet, plaque



class 495: china cabinet, china closet



class 500: cliff dwelling

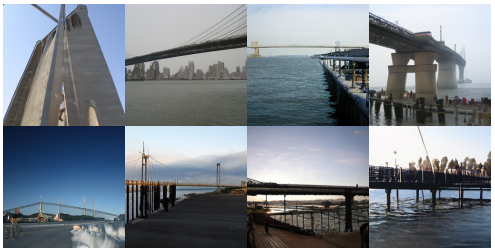
Figure 10: Qualitative samples on ImageNet 256×256, conditioned on the indicated classes, generated with JiT-H/16 + Dual Head and CFG of 2.8.



class 658: mitten



class 661: Model T



class 718: pier



class 724: pirate, pirate ship



class 725: pitcher, ewer



class 757: recreational vehicle, RV, R.V.



class 779: school bus



class 780: schooner



class 829: streetcar, tram, tramcar, trolley, trolley car



class 853: thatch, thatched roof

Figure 11: Qualitative samples on ImageNet 256×256, conditioned on the indicated classes, generated with JiT-H/16 + Dual Head and CFG of 2.8.


class 873: triumphal arch


class 900: water tower


class 911: wool, woolen, woollen


class 913: wreck


class 927: trifle


class 930: French loaf


class 946: cardoon


class 947: mushroom

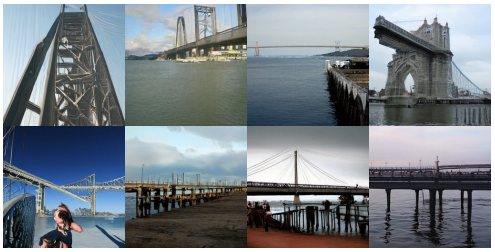

class 975: lakeside, lakeshore


class 989: hip, rose hip, rosehip

Figure 12: Qualitative samples on ImageNet 256×256, conditioned on the indicated classes, generated with JiT-H/16 + Dual Head and CFG of 2.8.


class 658: mitten


class 661: Model T


class 718: pier


class 724: pirate, pirate ship


class 725: pitcher, ewer


class 757: recreational vehicle, RV, R.V.


class 779: school bus


class 780: schooner


class 829: streetcar, tram, tramcar, trolley, trolley car


class 853: thatch, thatched roof

Figure 13: Qualitative samples on ImageNet 256×256, conditioned on the indicated classes, generated with JiT-L/16 + Dual Head and CFG of 2.6.