What kills v-prediction? A Patch-wise PCA Perspective on Pixel-Space Flow Matching

Tianze Luo[1,2], **Haofeng Huang**[1], **Yuan Yao**[1,2*]
[1]Tsinghua University, [2]Shanghai Qi Zhi Institute

Abstract

Latent Diffusion Models dominate generative modeling but face two-stage training complexity and the inherent reconstruction limits of pre-trained autoencoders. Pixel-space Flow Matching offers a simplified end-to-end alternative yet often encounters training difficulties when using the standard v-prediction objective. Recent research attributes this difficulty to the off-manifold nature of noised targets, motivating a shift toward x-prediction. In this paper, we investigate this explanation through patch-wise Principal Component Analysis (PCA) and identify a bandwidth bottleneck. The velocity in minor components is largely determined by the raw noisy input, requiring deep backbones to preserve information that competes with principal-component modeling. Through controlled experiments and representation analysis, we further examine why x-prediction remains effective: its parameterization provides a raw-input bypass, while its learned input representation can spontaneously concentrate on principal directions. Under low-rank observations, our analysis of the induced flow characterizes x-prediction as an implicit conditional-mean decoder, reconstructing discarded components from the retained information. These observations motivate the Raw-injected Transformer (RiT), which combines a lightweight raw-conditioned decoder with an output-side shortcut to separate semantic modeling from raw-state propagation. Our method restores effective v-prediction and achieves competitive performance on ImageNet 256×256, providing a perspective on the interaction between prediction objectives and architectural bandwidth. Our code is available at https://github.com/THUMAI-Lab/RiT.



(a) Curated samples from RiT-H/16 on ImageNet 256 × 256.

Size	JiT	RiT
B/16	3.66	**3.01**
L/16	2.36	**2.06**
H/16	1.86	**1.79**

(b) FID-50K comparison (↓)

Figure 1: **Qualitative results and quantitative comparison.** The visual examples illustrate the generation quality of our model, while the table compares FID-50K across different model scales on ImageNet 256 × 256. RiT-B and RiT-L are trained for 600 epochs, whereas RiT-H is trained for 480 epochs. All JiT [30] baselines are trained for 600 epochs. Our models consistently achieve lower FID-50K scores, even with fewer training epochs at the H scale.

1 Introduction

Diffusion and Flow Matching Models [1, 20, 32, 33, 51, 52] have emerged as a robust and powerful paradigm for visual generative modeling, demonstrating competitive generation quality and

*Corresponding author: yaoyuanthu@mail.tsinghua.edu.cn

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

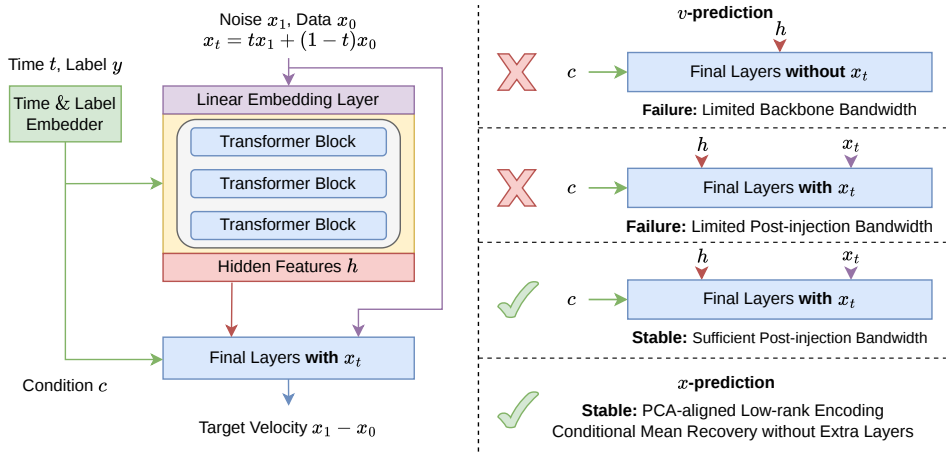


Figure 2: **Overview of RiT and the bandwidth bottleneck.** Standard v-prediction must retain noisy input while learning semantic features, creating competing demands on backbone bandwidth. Enlarging the backbone can alleviate this conflict without extra raw injection, but incurs substantial computational overhead. RiT instead uses a lightweight raw-conditioned nonlinear decoder with sufficient post-injection capacity and an output-side shortcut. Details of the final layers are given in Section 4.5. In contrast, x-prediction provides an inherent raw-input bypass and can learn compact representations concentrated on principal directions. Under low-rank conditions, its induced flow reconstructs discarded components as conditional means without additional decoder layers.

distribution coverage compared to Normalizing Flows and Generative Adversarial Networks (GANs) [2, 3, 11, 15–17, 19, 26, 35, 43–45, 48, 63]. While initially applied to image synthesis, this framework has been successfully extended to diverse high-dimensional domains, including video generation [28, 56, 59] and audio generation [7, 9, 13, 29, 34]. Although diffusion models originally operated in raw pixel space, their performance and training efficiency often degrade as image resolution increases. These difficulties arise in part from directly learning high-frequency spatial dependencies and the quadratic complexity of dense attention mechanisms commonly used in modern diffusion backbones.

To mitigate these computational constraints, Latent Diffusion Models (LDMs) [45] have become the dominant approach in recent years. By leveraging pre-trained Variational Autoencoders (VAEs) [27] to project inputs into a compressed, low-dimensional representation, LDMs effectively decouple generative modeling from high-resolution synthesis [36, 42]. However, the reliance on complex two-stage pipelines and the reconstruction ceiling of fixed decoders have recently spurred widespread research interest in end-to-end pixel-space diffusion models [8, 30, 62]. To maintain computational efficiency, these models typically adopt large patch sizes (e.g., 16×16 or 32×32), creating high-dimensional patch representations that pose new challenges for standard diffusion objectives.

Crucially, the recent JiT study [30] identifies a severe training failure in pixel space: while velocity prediction (v-prediction) is Flow Matching's native regression objective, applying it directly to high-dimensional pixel patches can fail. JiT therefore adopts clean-data prediction (x-prediction), which substantially improves generation quality in this setting. However, the cause of this failure remains incompletely understood. This difference is often attributed to the low-dimensional manifold nature of natural images. Yet the MSE-optimal x-predictor estimates a conditional expectation, which need not lie on the data manifold. The manifold interpretation alone therefore leaves open how prediction objectives interact with the information constraints of practical architectures.

We investigate this question by analyzing image patches through Principal Component Analysis (PCA). We observe that v-prediction converges in low-dimensional principal subspaces but deteriorates as minor components are introduced. Linear reconstruction probing reveals that v-prediction retains substantially more raw noisy-input information through the backbone than x-prediction. Our affine analysis links this behavior to the raw-state transport required by noise-dominated minor components, which competes with principal-component modeling. The x-prediction parameterization provides a raw-input bypass that alleviates this demand. Under low-rank observations, its induced flow completes discarded coordinates through conditional means without propagating their noisy states through the backbone. We further observe that even without an imposed low-rank bottleneck, its input embedding can spontaneously reduce its effective rank and concentrate on principal directions.

To resolve this feature bandwidth conflict, we propose the Raw-injected Transformer (**RiT**) for pixel-space Flow Matching. RiT injects the raw noisy input x_t into a lightweight subpatch decoder and combines a semantic branch with an explicit raw-input shortcut through a shared output projection. The decoder provides sufficient post-injection bandwidth to model conditional variation beyond mean reconstruction, while the shortcut supplies learnable raw-state transport. Together, these components allow content-dependent processing of the full noisy input without requiring the deep backbone to preserve every coordinate. The final architecture requires neither a precomputed PCA basis nor auxiliary perceptual, adversarial, or frequency-domain losses.

We conduct extensive experiments on the ImageNet 256×256 [10] dataset at B, L, and H scales. Controlled interventions on backbone capacity, patch dimensionality, raw-input availability, and decoder bandwidth support the bandwidth interpretation. RiT restores effective v-prediction and achieves competitive FID scores compared to x-prediction baselines across training horizons. These results indicate that the observed failure is not an inherent limitation of velocity prediction, but depends on whether the architecture supplies the information paths required by the objective.

2 Related Work

Latent Diffusion Models Latent diffusion has become a standard paradigm for efficient and scalable image and video generation [3, 14, 36, 43, 45, 61]. Early systems use VAEs [27] for compact representations, while the transition from U-Net [46] to DiT [42] improves scalability and generation quality. Subsequent autoencoder designs explore higher compression ratios, channel-wise latent structures, and semantic alignment for downstream diffusion [4, 5, 56, 60]. RAE [64] pairs pretrained representation encoders such as DINOv2 [40] with ViT decoders [12], forming Transformer-based autoencoders with high-dimensional semantic latents and improving diffusion convergence and generation quality. However, autoencoder training often involves perceptual and adversarial losses, adding optimization overhead and complexity to the two-stage generation pipeline.

Pixel Diffusion Models Diffusion models were originally developed in pixel space [20, 51, 52]. Early high-resolution approaches include diffusion upsampling in ADM [11] and progressive super-resolution cascades in CDM [21]. SiD [23] and SiD2 [24] improve pixel-space generation through U-ViT architectures, adapted noise schedules, and loss weighting. PixelFlow [6] trains a unified DiT for multi-resolution Flow Matching, whereas RelayDiffusion [54] uses separate models for different scales. Inspired by implicit neural fields for scene reconstruction [39, 50], PixNerd [57] introduces neural-field decoding after the DiT backbone, achieving competitive results at patch size 16.

More recently, JiT [30] finds clean-image prediction more effective than direct velocity prediction for large-patch pixel-space Transformers and attributes this advantage to the low-dimensional data manifold. Other methods augment DiT for high-frequency detail modeling: PixelDiT [62] adds pixel-level Transformer blocks with linear token compression and expansion; DeCo [37] couples a lightweight pixel decoder with a frequency-aware Flow Matching loss; and DiP [8] introduces a lightweight convolutional U-Net Patch Detailer Head. DDT [58] also separates semantic encoding from velocity decoding by supplying the noisy input directly to a dedicated decoder, but operates in compressed VAE latent space rather than on high-dimensional pixel patches.

3 Preliminary: Flow Matching

We review the principles of Flow Matching, a generative framework that transports a tractable prior to a target data distribution. Let $p(x_0)$ denote the data distribution and $p(x_1)$ represent the prior, typically defined as a standard Gaussian $\mathcal{N}(0, I)$. x_1 is sampled independently of x_0. The transition between a data sample x_0 and a noise sample x_1 is defined via linear interpolation, creating a time-dependent state $x_t = (1-t)x_0 + tx_1 \in \mathbb{R}^n$ for $t \in [0, 1]$. This probability path is governed by a marginal velocity field, given by the conditional expectation of the vector from x_0 to x_1:

$$v(x_t, t) = \mathbb{E}[x_1 - x_0 \mid x_t, t]. \tag{1}$$

To generate data samples, we first sample $x_1 \sim p(x_1)$ and then solve the corresponding ordinary differential equation (ODE) backward from $t = 1$ to $t = 0$:

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = v(X(t), t), \qquad X(1) = x_1. \tag{2}$$

In practice, the velocity field is parameterized by a neural network v_θ, and numerical methods such as Euler or Heun are used to approximate the trajectory. Direct regression against the marginal field is generally intractable because its conditional expectation is unavailable. The Conditional Flow Matching formulation [32, 33] provides a tractable objective with the same optimization gradients by using individual data-noise pairs:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,x_0,x_1} \left\| v_\theta(x_t, t) - (x_1 - x_0) \right\|_2^2 . \tag{3}$$

Its optimal predictor recovers the marginal velocity field. Recent works [30, 38] demonstrate that direct data prediction (x-prediction) is beneficial for pixel-space generation. Under this formulation, the network $\hat{x}_\theta$ regresses x_0, and the objective is re-weighted as

$$\mathcal{L}_{\mathrm{Data}}(\theta) = \mathbb{E}_{t,x_0,x_1} \left[1/\bar{t}^2 \left\| \hat{x}_\theta(x_t, t) - x_0 \right\|_2^2 \right], \qquad \bar{t} = \max(t, 0.05). \tag{4}$$

The time-dependent denominator is thresholded to prevent numerical instability near $t = 0$. Without clipping, the velocity is recovered through $v_\theta(x_t, t) = (x_t - \hat{x}_\theta(x_t, t))/t$ for $t > 0$. Our implementation uses $\bar{t}$ for the denominator in this expression as well. We distinguish the unclipped continuous-time relation from its practical implementation in the analysis below. Class conditioning is omitted here for clarity and denoted by ℓ where needed.

4 Methodology

4.1 Overview

We present our methodology for reviving v-prediction in pixel-space Flow Matching. Despite its simple, time-independent regression target $x_1 - x_0$, direct v-prediction can fail when high-dimensional pixel patches must be processed within a limited hidden representation. We investigate this difficulty through patch-wise PCA, which shows that v-prediction trains successfully in lower-dimensional principal subspaces but deteriorates as minor components are introduced. Linear reconstruction probing then reveals a stronger tendency to retain noisy-input information throughout the backbone.

We explain this behavior through an affine analysis of component-wise velocity transport and examine two complementary advantages of x-prediction: its raw-input bypass and its tendency to learn representations concentrated on principal directions. Under low-rank observations, we characterize how the induced flow reconstructs discarded components as conditional means. Building on these insights, RiT combines a lightweight raw-conditioned nonlinear decoder with an output-side shortcut. Controlled experiments on backbone width, patch dimensionality, decoder capacity, and raw-input availability test the bandwidth interpretation and clarify the roles of these components.

4.2 Patch-wise Principal Components Analysis

To understand the failure of v-prediction in pixel space, we first investigate the distribution of image patches. Natural images are highly redundant, and much of their centered variation is concentrated in a low-dimensional subspace. Let $\mu = \mathbb{E}[x_0] \in \mathbb{R}^p$ and $\Sigma = \mathrm{Cov}(x_0) \in \mathbb{R}^{p \times p}$ denote the patch mean and covariance, where p is the patch dimension. For a patch size of 16, $p = 3 \times 16 \times 16 = 768$. By performing eigendecomposition $\Sigma = V \Lambda V^\top$, we obtain the orthonormal eigenbasis $V = [v_1, \ldots, v_p]$ and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$, with $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$. Leading eigenvectors capture the dominant centered variation, while the remaining lower-variance directions are referred to as minor components. We define the affine projection onto the principal k-dimensional subspace as

$$P_k = V_k V_k^\top, \qquad \mathcal{P}_k^\mu(x) = \mu + P_k(x - \mu), \tag{5}$$

where V_k consists of the first k principal eigenvectors. We apply this map to both data and noise, so that both endpoints lie in $\mu + \mathrm{range}(P_k)$. This changes the stochastic dimensionality while preserving the original patch shape, and the map becomes the identity when $k = p$. Our implementation follows the JiT-B backbone [30], with hidden size 768 and approximately 132M parameters in these experiments. The intermediate width of its two-layer linear input embedding is also 768, so no architectural low-rank bottleneck is imposed here.

All experiments are conducted on ImageNet [10] at 256×256 resolution with patch size 16. We employ AdamW with learning rate 2×10^{-4}, zero weight decay, and betas $(0.9, 0.95)$. Training lasts 250,000 updates with global batch size 1024, including a linear warmup over the initial 6250 updates

and a constant learning rate thereafter. Evaluation uses an Exponential Moving Average (EMA) decay of 0.9996 and Fréchet Inception Distance (FID-50K) [18]. We use 50-step Heun sampling with a final Euler step, corresponding to 99 velocity evaluations. Classifier-Free Guidance (CFG) [22] is disabled to examine generation quality without guidance.

	v-prediction					x-prediction				
PCA Dim k	128	192	256	384	768 (Full)	128	192	256	384	768 (Full)
FID-50K (↓)	**33.99**	36.49	38.75	47.41	149.07 (Fail)	35.01	37.56	39.58	40.83	40.39

Table 1: **Effect of PCA projection.** While v-prediction fails in the full 768-dimensional pixel space, restricting both data and noise to lower-dimensional principal subspaces not only restores effective training but even allows v-prediction to outperform x-prediction at $k = 128, 192, 256$. Its performance then deteriorates as additional minor directions are introduced.

As reported in Table 1, the two objectives behave differently as the subspace dimension increases. In the 128-dimensional principal subspace, v-prediction converges successfully and achieves lower FID than x-prediction. At $k = 128, 192, 256$, training in the principal subspace also improves FID relative to full-space training for both objectives at this training budget. However, v-prediction is more sensitive to minor components: as the dimension expands from 128 to 384, its FID rises from 33.99 to 47.41, and full-space training deteriorates to 149.07. In contrast, x-prediction remains effective across all tested dimensions, reaching an FID of 40.39 at full rank. These observations suggest that x-prediction has an advantage in handling noise-dominated minor directions, and that the failure of direct v-prediction depends on the information demands placed on the architecture rather than solely on the velocity target itself.

4.3 The Bandwidth Bottleneck and Noisy Input Memorization

To uncover why processing minor components leads to difficulty, we analyze information flow within the network. Specifically, we investigate whether hidden states in deep layers retain the noisy input x_t. We perform linear probing to reconstruct x_t from the hidden states h_l at each layer l, measuring fidelity through the coefficient of determination (R^2). After training in the previous section, we fit one affine map per layer on a fixed set of analysis batches:

$$R^2(h_l, x_t) = 1 - \frac{\min_{W,b} \sum_n \|x_{t,n} - (W h_{l,n} + b)\|_2^2}{\sum_n \|x_{t,n} - \bar{x}_t\|_2^2}, \tag{6}$$

where n indexes image patches across the batches, and $\bar{x}_t$ is the empirical mean of the noisy input patches. All models are evaluated on the same images and noise draws, with the corresponding PCA projection applied to subspace models. Figure 3 reveals that v-prediction retains substantially more input information than x-prediction. This behavior is consistent with its target $x_1 - x_0$, which requires the current noisy state to remain available for minor-component velocity estimation.

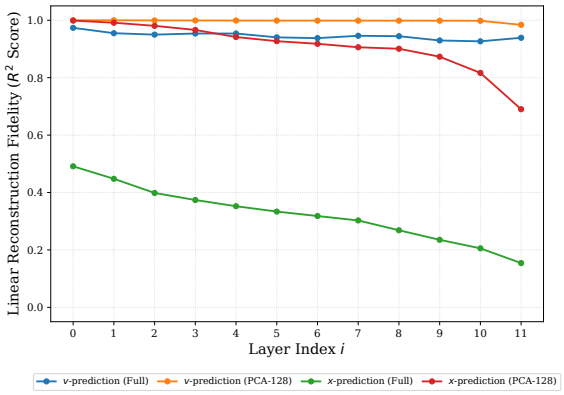


Figure 3: **Linear reconstruction fidelity (R^2 score) of x_t from h_l.** We observe that v-prediction tends to retain substantially more noisy-input information in the residual path than x-prediction, consistent with the raw-state contribution required for minor-component velocity estimation. When the patch and hidden dimensions are comparable, preserving this 768-dimensional state can compete with the capacity needed for principal-component modeling. In contrast, the PCA-128 input has fewer stochastic dimensions, allowing high input retention alongside effective generative learning.

In full pixel space, v-prediction maintains high reconstruction fidelity, with $R^2 \approx 0.95$ across much of the backbone. When the input dimensionality is comparable to the hidden width, preserving this

high-dimensional state can compete with semantic modeling. In contrast, the linear accessibility of x_t decreases with depth under x-prediction. Models trained in the PCA-128 subspace also show high input retention, yet do not fail: their 768-dimensional hidden states can accommodate a substantially smaller stochastic input while retaining capacity for nonlinear modeling. These probes indicate that substantially more noisy-input information remains linearly accessible in the hidden states under v-prediction, consistent with the increased raw-state routing demands associated with minor components. This connects the degradation in Section 4.2 to a bandwidth conflict between preserving high-dimensional noisy state and modeling principal components, which is further supported by the width, patch-size, and raw-access interventions in Section 4.5.

4.4 The Implicit Advantage of x-Prediction on Minor Components

Sections 4.2 and 4.3 show that x-prediction remains effective while retaining less noisy-input information. We now examine how its parameterization reduces the burden of raw-state propagation and permits reconstruction from low-rank observations. Throughout, $\widetilde{x}_t = x_t - (1-t)\mu$ and $\mu_i = v_i^\top \mu$ account for the patch mean.

Proposition 1 (Affine raw-state transport)**.** *For fixed $t > 0$, the minimum-MSE affine estimator of $v_i^\top (x_1 - x_0)$ from $v_i^\top x_t$ is $\widehat{V}_i^{\text{aff}} = -\mu_i + C_i(t) v_i^\top \widetilde{x}_t$, with*

$$C_i(t) = \frac{t - (1-t)\lambda_i}{t^2 + (1-t)^2 \lambda_i}, \qquad \operatorname{std}(r_i(t)) = \frac{\sqrt{\lambda_i}}{\sqrt{t^2 + (1-t)^2 \lambda_i}}, \tag{7}$$

where $r_i(t) = \widehat{V}_i^{\text{aff}} - v_i^\top (x_1 - x_0)$ measures error against the paired training target.

Figure 4 shows that many minor directions have very small eigenvalues. At fixed $t > 0$, $C_i(t) \to 1/t$ and the residual variance vanishes as $\lambda_i \to 0$, so the affine estimate approaches $(v_i^\top x_t - \mu_i)/t$. In these low-variance directions, the MSE-optimal clean-data prediction is close to the mean μ_i. Thus, the x-prediction parameterization $\hat{v}_\theta = (x_t - \hat{x}_\theta)/t$ provides approximately the same affine transport away from the low-noise endpoint. This reduces the backbone's transmission burden and motivates a learnable shortcut for v-prediction in Section 4.5. Appendix A provides the derivation. Importantly, the full x-prediction identity remains exact beyond this small-variance approximation.

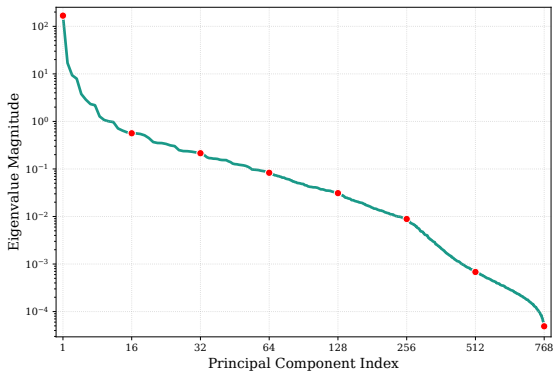


Figure 4: **Eigenvalue distribution of image patches.** Eigenvalues of the centered 768×768 patch covariance on the ImageNet 256×256 training set, shown in descending order. Their magnitudes span more than six orders, reflecting a strongly anisotropic distribution. Minor directions contain little data variance but receive unit-variance Gaussian noise in full-space training. Predicting their velocity therefore requires access to the noisy state even when the remaining regression error is small, motivating a raw-input shortcut that reduces this transmission burden.

This affine advantage does not fully explain why x-prediction remains effective with low-rank linear input embeddings, such as the 128-dimensional bottlenecks used for 768-dimensional patches in JiT-B/16 and JiT-L/16 [30]. Unlike the PCA intervention in Section 4.2, the full data and noise distributions are retained, but at least 640 directions per patch are unobserved by the backbone. The following result explains how x-prediction completes these discarded clean-data coordinates.

Proposition 2 (Conditional-mean completion)**.** *Let P_1 project orthogonally onto the row space of the linear input map W, and let $P_2 = I - P_1$ project onto its nullspace. Suppose the predictor observes only $Y_t = P_1 \widetilde{x}_t$, time, and class ℓ. Assume the retained flow is learned correctly and has a unique trajectory with an endpoint, and the minor conditional mean extends continuously along it to $t = 0$. The MSE-optimal x-predictor with the unclipped velocity parameterization then satisfies*

$$P_2 \widehat{X}(0) = \mathbb{E}[P_2 x_0 \mid P_1 x_0 = P_1 \widehat{X}(0), \ell]. \tag{8}$$

The integrating-factor proof in Appendix B shows that x-prediction removes the initial noise in unobserved directions and completes their clean-data values through conditional means, which may

depend nonlinearly on retained information. Direct v-prediction with the same observation instead retains unit-variance minor noise. The appendix also analyzes the effect of time clipping along a fixed retained trajectory. Given its ability to complete discarded coordinates, we next examine whether x-prediction also learns more compact input representations without an explicit input bottleneck. Figure 5 shows that its full-width input embedding spontaneously develops a lower effective rank and aligns with principal directions. For the composed map $W = W_2W_1$, we quantify this behavior through its singular-value entropy effective rank [47] and its operator energy in principal directions:

$$\mathrm{erank}(W) = \exp\left(-\sum_i \pi_i \log \pi_i\right), \quad \pi_i = \frac{\sigma_i(W)}{\sum_j \sigma_j(W)}, \quad E_K(W) = \frac{\|WV_K\|_F^2}{\|W\|_F^2}. \tag{9}$$

We fix $K = 280$ to approximately match the effective rank of full-space x-prediction after 200 epochs and assess PCA alignment using unit-norm PCA directions without eigenvalue weighting. Figure 5 shows that even with a full-width input embedding, x-prediction spontaneously reduces its effective rank and concentrates on principal directions, whereas v-prediction retains a broader spectrum. Reduced effective rank does not itself imply exact information loss, but this observation supports the relevance of the low-rank analysis.

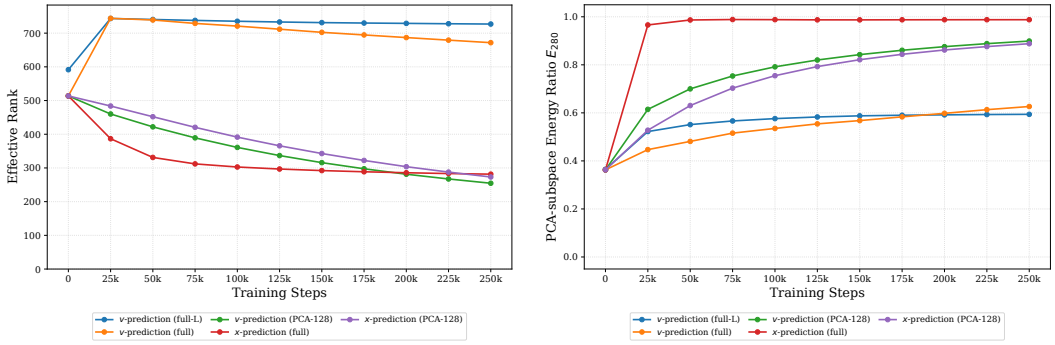


Figure 5: **Evolution of the learned input embedding.** Left: effective rank of $W = W_2W_1$, computed from the entropy of its normalized singular values to characterize the dimensionality of the learned representation. Right: energy in the first 280 PCA directions. Full-space x-prediction concentrates on principal directions, whereas v-prediction retains a broader spectrum. PCA-128 denotes projected data and noise, not an embedding bottleneck.

Conditional-mean completion captures nonlinear dependencies on retained information, but does not generally recover the joint conditional distribution of discarded components. This motivates a lightweight nonlinear decoder with direct raw-input access and sufficient bandwidth to model the remaining conditional variation. Combined with the learnable shortcut, this design improves FID over the x-prediction baseline in our experiments, as detailed in Section 4.5.

4.5 Raw-Conditioned Decoding Revives v-Prediction

Following this analysis, RiT combines a lightweight raw-conditioned decoder for modeling conditional variation with a learnable output shortcut for raw-state transport under direct v-prediction. Let s_b and s_d denote the backbone and decoder patch sizes, respectively, with s_b/s_d an integer.

Decoder bandwidth and raw-input access. Each backbone token is linearly expanded and rearranged into $(s_b/s_d)^2$ decoder tokens. Let h^{back} denote the backbone features, $x_t^{(s_d)}$ the raw decoder patches, and c the decoder condition. The resulting features are

$$h = \mathcal{D}_\theta\Big(\mathrm{split}(H_{\mathrm{in}}h^{\mathrm{back}}) + X_{\mathrm{proj}}x_t^{(s_d)},\ c\Big), \tag{10}$$

where $\mathcal{D}_\theta$ denotes the decoder Transformer and c is the projected sum of time and class embeddings. In our implementation, $s_b = 16$, $s_d = 8$, and $\mathcal{D}_\theta$ contains two blocks. The finer sequence is processed only by the decoder, while the deep backbone retains its original patch resolution. Each raw RGB decoder patch contributes $3s_d^2$ coordinates, which must be processed alongside the backbone features within the decoder's hidden representation.

Intervention	Backbone width	Patch	Decoder width	Raw input	FID-50K (↓)
Vanilla B	768	16	–	No	149.07
Vanilla L	1024	16	–	No	33.43
Vanilla B, smaller patches	768	8	–	No	28.70
Two-block decoder	768	16	128	Yes	220.78
Two-block decoder	768	16	256	Yes	34.68
Two-block decoder	768	16	384	No	186.16
Two-block decoder	768	16	384	Yes	31.85

Table 2: **Interventions on bandwidth and raw access.** All rows use v-prediction, 250K updates, and no guidance. Vanilla models have no low-rank input bottleneck. Decoder models use a rank-128 backbone input bottleneck and patch-8 decoder tokens, without an output-side shortcut. "Raw input" denotes an additional decoder input, not the ordinary backbone input.

As shown in Table 2, increasing backbone capacity or reducing patch size can restore vanilla v-prediction, but both interventions introduce additional computational costs. In particular, a patch-8 backbone increases the sequence length throughout the network, whereas RiT processes finer-resolution tokens using only two lightweight decoder blocks. Within the decoder family, two width-384 blocks fail without direct raw-input access but converge when it is provided. Conversely, a width-128 decoder fails despite receiving the 192-dimensional raw subpatches, while widths 256 and 384 train effectively. These controlled interventions suggest that both raw-input availability and sufficient post-injection bandwidth are important for restoring v-prediction.

A shared-output transport shortcut. The nonlinear decoder must still learn the simple raw-state contribution identified by the affine analysis. We provide a short output-side path for this term and combine it with semantic features before a shared output projection. For decoder width d, a SiLU followed by a linear layer maps the condition c to five channel-wise coefficients:

$$[c_1, c_2, c_3, c_4, c_5] = \mathrm{Linear}(\mathrm{SiLU}(c)), \qquad c_i \in \mathbb{R}^d. \tag{11}$$

Following the coefficient numbering in the implementation, the raw and semantic features are

$$r = X_{\mathrm{in}} x_t^{(s_d)}, \qquad q = H_{\mathrm{out}}(\mathrm{RMSNorm}(h) \odot (1 + c_4) + c_5)\,. \tag{12}$$

The fractional form is inspired by the rational time dependence of the optimal affine coefficient, which satisfies $C_i(1) = 1$ at the pure-noise endpoint. We anchor the raw gain to one there and adopt the same endpoint convention for the semantic gain:

$$d_t = \max\{t + (1-t)c_2, \epsilon\}, \qquad g_r = \frac{1 + (1-t)c_1}{d_t}, \qquad g_h = \frac{1 + (1-t)c_3}{d_t}, \tag{13}$$

with $\epsilon = 0.05$, giving the final prediction

$$\hat{v}_\theta = \mathrm{unpatchify}(X_{\mathrm{out}}(g_r \odot r + g_h \odot q))\,. \tag{14}$$

All divisions and denominator clipping operations are applied element-wise, and coefficients are broadcast over subpatch positions. The projections are learned affine maps. X_{in} is distinct from the decoder injection X_{proj}, whereas X_{out} is shared by the raw and semantic branches. H_{out} maps semantic features into the common output coordinates before their scaled contributions are combined.

The endpoint constraints apply to the modulation gains rather than the entire velocity field, since the projections and semantic contribution remain learnable. Neither PCA coordinates nor exact conditional variances are prescribed. We initialize H_{out} and the coefficient outputs to zero, initialize X_{in} with orthonormal columns, and initialize X_{out} with its transposed weight, using zero biases. For the final decoder widths, the initial predictor is therefore $x_t/\bar{t}$. The transpose relation is not enforced during training, and the nonlinear decoder learns alongside the shortcut without requiring a precomputed PCA basis. We compare several output formulations in Table 3. The basic shortcut uses directly learned gains, the free fractional variant omits the $(1-t)$ factors, and the separate-denominator variant predicts independent denominators for the raw and semantic branches.

Their exact coefficient assignments are given in Appendix C.2. Even the basic shortcut improves FID over the standard decoder readout. The shared anchored form achieves the best FID among the tested variants, while additional denominator freedom does not improve this comparison. The same architecture also supports x-prediction, but direct v-prediction achieves lower FID under the common recipe. These results complement the bandwidth interventions by showing the benefit of a short learnable transport path alongside raw-conditioned nonlinear decoding.

Output shortcut	Objective	FID-50K (↓)
None (standard decoder readout)	v	31.85
Basic learned gains	v	28.27
Free fractional, shared denominator	v	28.33
Anchored fractional, separate denominators	v	28.51
Anchored fractional, shared denominator (RiT)	v	**27.97**
Anchored fractional, shared denominator	x	28.49

Table 3: **Output-shortcut ablation.** All models use the B-scale backbone, a rank-128 input bottleneck, and a raw-conditioned two-block width-384 decoder. Results are measured after 250K updates without guidance. The output-side shortcut improves FID over the standard decoder readout, with the anchored fractional parameterization performing best among the tested variants. The x-prediction row uses the same architecture with the clean-data objective to compare the two prediction targets.

5 Additional Experiments

Input bottleneck and time sampling. We further investigate architectural and training hyperparameters using RiT-B, with results summarized in Table 4. For the input bottleneck, we vary the intermediate width of the linear backbone embedding from 128 to 768 while keeping the two-block, 384-dimensional decoder unchanged. Reducing the bottleneck width consistently improves FID, with the 128-dimensional setting achieving the best result. Unlike the PCA intervention in Section 4.2, these experiments retain the full data and noise distributions and constrain only the learned backbone input map. The bottleneck therefore affects only the backbone embedding, leaving the decoder's separate linear projection of the full raw input unchanged.

Regarding time sampling, we draw $\mathrm{logit}(t) \sim \mathcal{N}(\mu_t, \sigma_t^2)$ and evaluate four choices of (μ_t, σ_t). Increasing μ_t shifts the sampling distribution toward higher noise levels and improves FID. Further reducing σ_t from 1.0 to 0.8 at $\mu_t = 0.8$ yields the best result. Together, these observations are consistent with the design choices in JiT [30]: once the routing bottleneck is alleviated, direct v-prediction also benefits from a low-rank input embedding and a noise-biased training distribution.

Bottleneck Width	128	256	384	768
FID-50K (↓)	**27.97**	29.63	29.70	30.45

(μ_t, σ_t)	(0.0, 1.0)	(0.4, 1.0)	(0.8, 1.0)	(0.8, 0.8)
FID-50K (↓)	33.69	29.71	29.48	**27.97**

Table 4: **Hyper-parameter ablation.** Following the x-prediction baseline [30], we find that a lower-rank input bottleneck and a high-noise sampling bias consistently enhance performance. All results are obtained with RiT-B after 200 epochs without classifier-free guidance.

Generation quality across model scales. To evaluate scalability, we train RiT-B/16 and RiT-L/16 for approximately 600 epochs and RiT-H/16 for approximately 480 epochs. These models build on the corresponding JiT Transformer backbones with our lightweight raw-conditioned decoders. Detailed training and inference configurations are provided in Appendix C, including the late-stage shortcut learning-rate reduction for RiT-H. All models operate directly in pixel space without PCA projection or auxiliary perceptual, adversarial, frequency-domain, or representation-alignment losses.

Table 5 summarizes the generative performance on ImageNet 256×256. RiT-B/16 and RiT-L/16 achieve FID scores of 3.01 and 2.06 after 600 epochs, compared with the reported JiT results of 3.66 and 2.36. RiT-H/16 reaches 1.79 after 480 epochs, while JiT-H/16 reports 1.86 after 600 epochs. These results support our analysis that v-prediction need not be inferior to x-prediction in pixel space. With sufficient bandwidth and direct raw-input access, v-prediction becomes effective across model scales. The previously observed failure therefore reflects a mismatch between the objective's information requirements and the architecture, not an inherent limitation of velocity prediction.

Performance at 200 epochs. The improvement is evident early in training. At 250K updates, corresponding to approximately 200 epochs, RiT-B/16, RiT-L/16, and RiT-H/16 achieve FID scores of 3.40, 2.48, and 2.17, respectively. These results improve upon the 4.37, 2.79, and 2.29 reported by JiT for the corresponding backbone scales at 200 epochs [30]. Detailed EMA and guidance configurations for these intermediate checkpoints and the final models are provided in Appendix C.3.

Method	Epochs	Params	IS (↑)	FID-50K (↓)
Generative Adversarial Networks				
BigGAN [2]	-	112M	171.4	6.95
StyleGAN-XL [49]	-	166M	**265.1**	**2.30**
GigaGAN [26]	-	569M	225.5	3.45
Masked and Autoregressive Models				
LlamaGen-XL [53]	300	775M	244.1	2.62
VAR-*d30* [55]	350	2.0B	**323.1**	1.92
MAR [31]	800	943M	303.7	**1.55**
RandAR-XL [41]	300	775M	314.2	2.22
Latent Diffusion Models				
LDM-4-G [45]	170	400M	247.7	3.60
DiT-XL/2 [42]	1400	675M	278.2	2.27
SiT-XL/2 [36]	1400	675M	277.5	2.06
LightningDiT-XL [60]	800	675M	295.3	1.35
REPA [61], SiT-XL/2	800	675M	**305.7**	1.42
RAE [64], DiT$^{\text{DH}}$-XL	800	839M	262.6	**1.13**
Pixel Space Diffusion Models				
ADM-G [11]	400	554M	186.7	4.59
RIN [25]	480	410M	182.0	3.42
PixelFlow-XL/4 [6]	320	677M	282.1	1.98
PixNerd-XL/16 [57]	160	700M	297.0	2.15
DiP-XL/16 [8]	600	631M	281.9	1.79
DeCo-XL/16† [37]	600	682M	**304.0**	**1.69**
JiT-B/16 [30]	600	131M	275.1	3.66
RiT-B/16	600	137M	300.3	3.01
JiT-L/16 [30]	600	459M	298.5	2.36
RiT-L/16	600	470M	331.4	2.06
JiT-H/16 [30]	600	953M	303.4	1.86
RiT-H/16	480	979M	**332.9**	**1.79**

Table 5: **Class-conditional ImageNet** 256×256 **results.** Baselines follow their own reported sampling settings. †DeCo additionally employs frequency-aware and representation-alignment losses.

Our contribution is primarily explanatory and methodological. We focus on understanding the information requirements of pixel-space prediction objectives and providing an architectural intervention that alleviates the resulting bandwidth bottleneck. This perspective complements other advances in generative modeling, including representation alignment (REPA [61]), in-patch convolutional decoders (DiP [8]), and frequency-aware training losses (DeCo [37]). RiT provides a lightweight realization of the design principles identified by our analysis, rather than prescribing a single architecture for pixel-space generation. These results suggest ways to build on these principles through more expressive raw-conditioned decoders and complementary training techniques.

6 Conclusion

We investigate the failure of direct v-prediction in pixel-space Flow Matching through patch-wise PCA, affine analysis, linear reconstruction probes, and controlled architectural experiments. Our results identify a bandwidth bottleneck in which preserving raw noisy state for minor-component velocity estimation competes with principal-component modeling. We further examine two complementary aspects of x-prediction: its parameterization provides a raw-input bypass, while its learned input embedding can spontaneously concentrate on principal directions. Under low-rank observations, our flow analysis characterizes x-prediction as an implicit conditional-mean decoder, completing discarded components without propagating their noisy states through the backbone. This completion can capture nonlinear dependence on retained information but does not generally recover the full conditional distribution. RiT combines a lightweight raw-conditioned nonlinear decoder for modeling conditional variation with a learnable output shortcut for raw-state transport, achieving competitive performance on ImageNet 256×256. These findings show that v-prediction can be effective when architectural bandwidth and information paths accommodate the demands of the prediction objective.

Acknowledgment and Funding Transparency Statement

This work was supported by the Shanghai Qi Zhi Institute Innovation Program (SQZ202410). We gratefully acknowledge the Shanghai Qi Zhi Institute for providing access to NVIDIA A800 GPUs for our experiments. Beyond the institutional affiliations and financial and computational support disclosed in this paper, the authors have no additional competing interests to declare.

References

[1] Michael Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *ICLR 2023 Conference*, 2023.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019.

[3] Huanqia Cai, Sihan Cao, Ruoyi Du, Peng Gao, Aiming Hao, Steven Hoi, Zhaohui Hou, Shijie Huang, Dengyang Jiang, Yuming Jiang, et al. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

[4] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv preprint arXiv:2410.10733*, 2024.

[5] Junyu Chen, Dongyun Zou, Wenkun He, Junsong Chen, Enze Xie, Song Han, and Han Cai. Dc-ae 1.5: Accelerating diffusion model convergence with structured latent space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19628–19637, 2025.

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

[7] Yushen Chen, Zhikang Niu, Ziyang Ma, Keqi Deng, Chunhui Wang, Jian Zhao, Kai Yu, and Xie Chen. F5-tts: A fairytaler that fakes fluent and faithful speech with flow matching. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6255–6271, 2025.

[8] 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. *arXiv preprint arXiv:2511.18822*, 2025.

[9] Ho Kei Cheng, Masato Ishii, Akio Hayakawa, Takashi Shibuya, Alexander Schwing, and Yuki Mitsufuji. Mmaudio: Taming multimodal joint training for high-quality video-to-audio synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28901–28911, 2025.

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

[11] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in neural information processing systems*, volume 34, pages 8780–8794, 2021.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[13] Zhihao Du, Qian Chen, Shiliang Zhang, Kai Hu, Heng Lu, Yexin Yang, Hangrui Hu, Siqi Zheng, Yue Gu, Ziyang Ma, et al. Cosyvoice: A scalable multilingual zero-shot text-to-speech synthesizer based on supervised semantic tokens. *arXiv preprint arXiv:2407.05407*, 2024.

[14] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[15] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *ICLR*, 2025.

[16] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *NeurIPS*, 2025.

[17] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[18] 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. In *Advances in neural information processing systems*, volume 30, 2017.

[19] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International conference on machine learning*, pages 2722–2730. PMLR, 2019.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in neural information processing systems*, volume 33, pages 6840–6851, 2020.

[21] 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.

[22] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

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

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

[25] Allan Jabri, David J Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In *ICML*, 2023.

[26] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *CVPR*, 2023.

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

[28] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

[29] Sang-Hoon Lee, Ha-Yeong Choi, and Seong-Whan Lee. Periodwave: Multi-period flow matching for high-fidelity waveform generation. In *International Conference on Learning Representations*, volume 2025, pages 30903–30933, 2025.

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

[31] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024.

[32] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] 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 (ICLR)*, 2023.

[34] Tianze Luo, Xingchen Miao, and Wenbo Duan. Wavefm: A high-fidelity and efficient vocoder based on flow matching. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2187–2198, 2025.

[35] Tianze Luo, Haotian Yuan, and Zhuang Liu. Soflow: Solution flow models for one-step generative modeling. In *International Conference on Learning Representations*, volume 2026, pages 101623–101641, 2026.

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

[37] Zehong Ma, Longhui Wei, Shuai Wang, Shiliang Zhang, and Qi Tian. Deco: Frequency-decoupled pixel diffusion for end-to-end image generation. *arXiv preprint arXiv:2511.19365*, 2025.

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

[39] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[40] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[41] Ziqi Pang, Tianyuan Zhang, Fujun Luan, Yunze Man, Hao Tan, Kai Zhang, William T Freeman, and Yu-Xiong Wang. Randar: Decoder-only autoregressive visual generation in random orders. In *CVPR*, 2025.

[42] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[43] Qi Qin, Le Zhuo, Yi Xin, Ruoyi Du, Zhen Li, Bin Fu, Yiting Lu, Xinyue Li, Dongyang Liu, Xiangyang Zhu, et al. Lumina-image 2.0: A unified and efficient image generative framework. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20031–20042. IEEE, 2025.

[44] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

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

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[47] Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007.

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

[49] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *SIGGRAPH*, 2022.

[50] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in neural information processing systems*, volume 33, pages 7462–7473, 2020.

[51] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in neural information processing systems*, volume 32, 2019.

[52] 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*, 2021.

[53] 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.

[54] Jiayan Teng, Wendi Zheng, Ming Ding, Wenyi Hong, Jianqiao Wangni, Zhuoyi Yang, and Jie Tang. Relay diffusion: Unifying diffusion process across resolutions for image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024.

[55] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *NeurIPS*, 2024.

[56] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

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

[58] 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*, pages 40633–40642, 2026.

[59] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

[61] 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 *ICLR*, 2025.

[62] Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. *arXiv preprint arXiv:2511.20645*, 2025.

[63] Shuangfei Zhai, Ruixiang Zhang, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Ángel Bautista, Navdeep Jaitly, and Joshua M Susskind. Normalizing flows are capable generative models. In *International Conference on Machine Learning*, pages 74348–74369. PMLR, 2025.

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

Appendix

A Mathematical Analysis of the Velocity Field in Minor Components

In this section, we derive the optimal time-dependent affine coefficient $C_i(t)$ and its residual standard deviation, proving Proposition 1. The calculation retains a nonzero patch mean and formalizes the raw-state transport required in noise-dominated directions. It explains the usefulness of the x_t/t term in x-prediction and motivates learning this contribution through an explicit shortcut.

A.1 Statistical Assumptions and Problem Setup

Let $x_0 \in \mathbb{R}^p$ denote an image patch with mean μ and covariance Σ. Its orthonormal eigenvectors $v_1, \ldots, v_p$ have eigenvalues $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$. Let $x_1 \sim \mathcal{N}(0, I)$ be independent Gaussian noise and $x_t = (1-t)x_0 + tx_1$. For a fixed $t > 0$, define

$$\mu_i = v_i^\top \mu, \qquad U_i = v_i^\top (x_0 - \mu), \qquad N_i = v_i^\top x_1. \tag{15}$$

Then $\mathbb{E}[U_i] = \mathbb{E}[N_i] = 0$, and independence gives

$$\mathbb{E}[U_i^2] = \lambda_i, \qquad \mathbb{E}[N_i^2] = 1, \qquad \mathbb{E}[U_i N_i] = 0. \tag{16}$$

Using the centered state $\widetilde{x}_t = x_t - (1-t)\mu$, write

$$Q_i(t) = v_i^\top \widetilde{x}_t = (1-t)U_i + tN_i, \qquad T_i = N_i - U_i. \tag{17}$$

The physical target coordinate is $v_i^\top (x_1 - x_0) = T_i - \mu_i$. We seek the affine estimator $a_i v_i^\top x_t + b_i$ minimizing

$$\mathcal{L}_i(a_i, b_i; t) = \mathbb{E}\left[\left(a_i v_i^\top x_t + b_i - v_i^\top (x_1 - x_0)\right)^2\right]. \tag{18}$$

For any a_i, minimizing over the intercept gives $b_i^* = -\mu_i - (1-t)a_i\mu_i$. The remaining problem is therefore

$$\mathcal{L}_i(a_i, b_i^*; t) = \mathbb{E}[(a_i Q_i(t) - T_i)^2]. \tag{19}$$

All expectations are taken at fixed t. Orthogonality separates the squared error along the axes; no independence between clean-data coordinates or Gaussian data assumption is required.

A.2 Derivation of the Optimal Coefficient $C_i(t)$

Taking the derivative with respect to a_i and setting it to zero yields

$$2\mathbb{E}[(a_i Q_i - T_i)Q_i] = 0, \qquad C_i(t) = \frac{\mathbb{E}[T_i Q_i]}{\mathbb{E}[Q_i^2]}. \tag{20}$$

The denominator follows by expanding the noisy coordinate and using the vanishing cross term:

$$\begin{aligned} \mathbb{E}[Q_i^2] &= (1-t)^2\mathbb{E}[U_i^2] + 2t(1-t)\mathbb{E}[U_i N_i] + t^2\mathbb{E}[N_i^2] \\ &= t^2 + (1-t)^2\lambda_i. \end{aligned} \tag{21}$$

Similarly, the numerator is

$$\begin{aligned} \mathbb{E}[T_i Q_i] &= \mathbb{E}[(N_i - U_i)((1-t)U_i + tN_i)] \\ &= t\mathbb{E}[N_i^2] - (1-t)\mathbb{E}[U_i^2] = t - (1-t)\lambda_i. \end{aligned} \tag{22}$$

Since the denominator is positive for $t > 0$, the optimal coefficient and the corresponding physical estimator are

$$C_i(t) = \frac{t - (1-t)\lambda_i}{t^2 + (1-t)^2\lambda_i}, \qquad \widehat{V}_i^{\text{aff}}(x_t, t) = -\mu_i + C_i(t) v_i^\top \widetilde{x}_t. \tag{23}$$

Thus, a nonzero mean changes the affine intercept without changing the covariance-dependent slope. This is the mean-corrected form of the component-wise transport analyzed in the main text.

A.3 Derivation of the Residual Standard Deviation

Define $r_i(t) = \widehat{V}_i^{\mathrm{aff}}(x_t, t) - v_i^\top(x_1 - x_0) = C_i(t)Q_i - T_i$. Its mean is zero. Expanding its variance and substituting the optimal coefficient gives

$$\begin{aligned}\mathrm{Var}(r_i) &= C_i^2\mathbb{E}[Q_i^2] - 2C_i\mathbb{E}[T_iQ_i] + \mathbb{E}[T_i^2] \\ &= \mathbb{E}[T_i^2] - \frac{\mathbb{E}[T_iQ_i]^2}{\mathbb{E}[Q_i^2]}.\end{aligned} \tag{24}$$

Independence implies $\mathbb{E}[T_i^2] = 1 + \lambda_i$. Substituting the moments derived above and collecting terms yields

$$\begin{aligned}\mathrm{Var}(r_i) &= (1+\lambda_i) - \frac{(t-(1-t)\lambda_i)^2}{t^2 + (1-t)^2\lambda_i} \\ &= \frac{(1+\lambda_i)(t^2 + (1-t)^2\lambda_i) - (t-(1-t)\lambda_i)^2}{t^2 + (1-t)^2\lambda_i} \\ &= \frac{\lambda_i\big(t^2 + 2t(1-t) + (1-t)^2\big)}{t^2 + (1-t)^2\lambda_i} = \frac{\lambda_i}{t^2 + (1-t)^2\lambda_i}.\end{aligned} \tag{25}$$

Taking the square root gives the residual standard deviation in Proposition 1:

$$\mathrm{std}(r_i(t)) = \frac{\sqrt{\lambda_i}}{\sqrt{t^2 + (1-t)^2\lambda_i}}. \tag{26}$$

The variance explained by the affine estimate is

$$\mathrm{Var}(\widehat{V}_i^{\mathrm{aff}}) = \frac{(t-(1-t)\lambda_i)^2}{t^2 + (1-t)^2\lambda_i}. \tag{27}$$

The estimate and its residual are uncorrelated by the least-squares normal equation, and their variances sum to $1 + \lambda_i$. The residual is measured against the paired training target; it may include irreducible conditional uncertainty and should not be interpreted entirely as a signal that nonlinear features can recover from the available noisy state.

A.4 Implications for Model Architecture and Information Flow

For fixed $t > 0$, taking $\lambda_i \to 0$ gives $C_i(t) \to 1/t$ and $\mathrm{std}(r_i(t)) \to 0$. Restoring the mean, the estimated physical velocity approaches

$$\widehat{V}_i^{\mathrm{aff}}(x_t, t) \longrightarrow -\mu_i + \frac{v_i^\top x_t - (1-t)\mu_i}{t} = \frac{v_i^\top x_t - \mu_i}{t}. \tag{28}$$

Thus, in directions with sufficiently small data variance, a scaled raw-state term explains most of the target variation. The x-prediction parameterization supplies x_t/t explicitly and uses its clean-data estimate to provide the remaining correction, including the nonzero mean. It can therefore reduce the need for the backbone to transmit each noisy coordinate through all its layers.

This argument concerns the linear contribution to the learning problem. The complete identity $(x_t - \mathbb{E}[x_0 \mid x_t, t, \ell])/t$ remains valid for arbitrary data distributions at $t > 0$; it does not assume that all eigenvalues vanish. The small-eigenvalue limit is also not uniform near the data endpoint: for every fixed $\lambda_i > 0$, $C_i(t) \to -1$ as $t \to 0$, while $C_i(1) = 1$. This supports a transport prior without making $1/t$ the optimal affine slope throughout the flow.

Direct v-prediction must obtain the same state-dependent contribution from its available features. When noisy-state propagation competes with principal-component modeling, a raw-input shortcut can provide this contribution without routing it through the deep backbone. RiT learns the shortcut's coefficients jointly with a nonlinear decoder rather than prescribing a PCA basis or fixed covariance model. The additional ability of x-prediction to discard input coordinates and complete them nonlinearly is a separate mechanism, developed in Appendix B.

B Low-rank Observations and Conditional-mean Reconstruction

In this section, we prove Proposition 2 using an integrating factor. The affine analysis in Appendix A explains the raw-state contribution to velocity estimation. Here, we examine how x-prediction handles information excluded by a low-rank input embedding. We retain the full data and noise distributions and allow nonlinear dependence between retained and discarded data components.

B.1 Low-rank Observations with a Nonzero Mean

Even with a full-width input embedding, Figure 5 shows that x-prediction spontaneously learns low-effective-rank representations concentrated on leading PCA directions. This motivates our analysis of low-rank observations, although a low effective rank does not necessarily imply exact information loss. Moreover, JiT-B/16 and JiT-L/16 use an input embedding bottleneck of width 128, while JiT-H/16 uses width 256 [30]. Our final RiT models retain these bottlenecks. For two consecutive linear maps $W_1 \in \mathbb{R}^{d_b \times p}$ and $W_2 \in \mathbb{R}^{d_h \times d_b}$, their composition $W = W_2 W_1$ satisfies

$$\operatorname{rank}(W) \le d_b, \qquad \dim \ker W = p - \operatorname{rank}(W) \ge p - d_b. \tag{29}$$

For a 768-dimensional patch, at least 640 directions are therefore unobservable to the B/L backbones, and at least 512 to H. For any $\delta \in \ker W$, the embeddings of x_t and $x_t + \delta$ are identical, so subsequent backbone layers cannot distinguish these inputs at fixed time and class. In a separate measurement of the rank-128 x-prediction model, over 99% of the embedding operator energy lies in the leading 128 PCA directions after 250K training steps, providing further evidence of alignment with principal components. The discarded directions need not coincide exactly with individual PCA axes. Under the MSE objective, an optimal x-predictor must estimate their clean-data values from the retained information, leading to the conditional-mean analysis below.

For the PCA-aligned case in Proposition 2, let $P_1 = P_k$ and $P_2 = I - P_1$. More generally, choose P_1 as the orthogonal projection onto the row space of the trained W, and P_2 onto its nullspace. The proof below applies to either choice. The observation restriction therefore applies without exact PCA alignment. The measured alignment explains why retained coordinates can be associated with principal directions in practice. Define $\mu_j = P_j \mu$ and the centered data and noise components

$$Y_0 = P_1(x_0 - \mu), \quad Z_0 = P_2(x_0 - \mu), \qquad Y_1 = P_1 x_1, \quad Z_1 = P_2 x_1. \tag{30}$$

Then, with $\widetilde{x}_t = x_t - (1-t)\mu$, the noisy coordinates are

$$Y_t = P_1 \widetilde{x}_t = (1-t)Y_0 + tY_1, \qquad Z_t = P_2 \widetilde{x}_t = (1-t)Z_0 + tZ_1. \tag{31}$$

For full images, the same notation denotes the corresponding block-diagonal projections across patches, the patch mean is repeated, and Y_t includes all retained tokens. We assume finite data second moments and standard Gaussian noise $x_1 \sim \mathcal{N}(0, I)$ independent of (x_0, ℓ), with time sampled independently. For the projected noise, zero mean and $P_j^\top = P_j$, $P_j^2 = P_j$, $P_1 P_2 = 0$ give the following covariances:

$$\begin{aligned} \operatorname{Cov}(Y_1, Z_1) &= \mathbb{E}[(P_1 x_1)(P_2 x_1)^\top] = P_1 \mathbb{E}[x_1 x_1^\top] P_2^\top = P_1 P_2 = 0, \\ \operatorname{Cov}(Z_1) &= \mathbb{E}[(P_2 x_1)(P_2 x_1)^\top] = P_2 \mathbb{E}[x_1 x_1^\top] P_2^\top = P_2. \end{aligned} \tag{32}$$

Because Y_1 and Z_1 are jointly Gaussian, zero cross-covariance implies independence. Since the whole noise vector is independent of (x_0, ℓ), Z_1 is independent of (Y_0, Z_0, Y_1, ℓ) and hence of (Y_t, t, ℓ). In an orthonormal basis of the discarded subspace, P_2 represents identity covariance. The clean-data components Y_0 and Z_0 may nevertheless be dependent. Expanding the conditional covariance of Z_t and using the same independence yields the exact relation

$$\operatorname{Cov}(Z_t \mid Y_t, t, \ell) = (1-t)^2 \operatorname{Cov}(Z_0 \mid Y_t, t, \ell) + t^2 P_2. \tag{33}$$

Thus, at $t > 0$, each discarded direction retains independent noise variance t^2 even if its clean value is predictable from the retained coordinates. This distinguishes loss of raw-state information from the possibility of predicting missing clean-data structure. When $WP_2 = 0$, the embedding satisfies

$$W x_t + b = W Y_t + (1-t) W \mu + b. \tag{34}$$

If W is injective on $\operatorname{range}(P_1)$, WY_t uniquely determines Y_t. In particular, this is automatic when P_1 projects onto the row space: a vector in that space with $WY_t = 0$ also lies in its orthogonal

complement $\ker W$ and must be zero. The known time-dependent offset can be subtracted, so observing (Wx_t+b, t, ℓ) is equivalent to observing (Y_t, t, ℓ). This equivalence concerns the backbone input, and RiT's additional raw paths restore information outside it.

Define the minor conditional mean

$$m_t(y, \ell) = \mathbb{E}[Z_0 \mid Y_t = y, t, \ell]. \tag{35}$$

For any estimate a, representing a predicted value of the discarded clean-data component $P_2 x_0$ based on the fixed observation (y, t, ℓ), subtracting and adding $\mu_2 + m_t(y, \ell)$ in the error gives

$$\begin{aligned}
&\mathbb{E}[\|a - P_2 x_0\|_2^2 \mid Y_t = y, t, \ell] && (36)\\
&= \|a - \mu_2 - m_t(y, \ell)\|_2^2 - 2(a - \mu_2 - m_t(y, \ell))^\top \mathbb{E}[Z_0 - m_t(y, \ell) \mid Y_t = y, t, \ell] && (37)\\
&\quad + \mathbb{E}[\|Z_0 - m_t(y, \ell)\|_2^2 \mid Y_t = y, t, \ell] && (38)\\
&= \|a - \mu_2 - m_t(y, \ell)\|_2^2 + \mathbb{E}[\|Z_0 - m_t(y, \ell)\|_2^2 \mid Y_t = y, t, \ell]. && (39)
\end{aligned}$$

The cross term vanishes because $\mathbb{E}[Z_0 - m_t(y, \ell) \mid Y_t = y, t, \ell] = 0$ by the definition of the conditional mean. Only the first term depends on a, so the optimal minor clean-data estimate is $\mu_2 + m_t(y, \ell)$. A positive time-dependent loss weight leaves this minimizer unchanged. Thus, a hard bottleneck guarantees an observation restriction, while the following endpoint result additionally assumes optimal regression and correctly learned retained dynamics.

B.2 Integrating-factor Solution of the x-Prediction Flow

The optimal retained clean-data estimate is $\mu_1 + \mathbb{E}[Y_0 \mid Y_t, t, \ell]$. After subtracting the time-dependent mean, its unclipped velocity induces the principal equation

$$y'(t) = \frac{y(t) - \mathbb{E}[Y_0 \mid Y_t = y(t), t, \ell]}{t}, \qquad y(1) = Y_1. \tag{40}$$

As in Proposition 2, assume this principal marginal flow is correctly learned and has a unique trajectory with an endpoint $y(0)$. Its right-hand side uses only the retained state, time, and class. The trajectory is therefore determined by (Y_1, ℓ) independently of Z_1, and need not follow an individual straight data-noise interpolation. Write the generated full trajectory as

$$\widehat{X}(t) = (1-t)\mu + y(t) + z(t), \qquad y(t) \in \operatorname{range}(P_1), \quad z(t) \in \operatorname{range}(P_2), \tag{41}$$

where $z(1) = Z_1$. Along this trajectory, abbreviate $m_t(y(t), \ell)$ as m_t. The minor part of the state is $u(t) = P_2 \widehat{X}(t) = (1-t)\mu_2 + z(t)$, and its predicted clean value is $\mu_2 + m_t$. The velocity parameterization consequently gives

$$u'(t) = \frac{u(t) - \mu_2 - m_t}{t} = -\mu_2 + \frac{z(t) - m_t}{t}. \tag{42}$$

Differentiating the definition of $u(t)$ gives $u'(t) = -\mu_2 + z'(t)$. Equating the two expressions cancels the mean terms and leaves

$$z'(t) - \frac{z(t)}{t} = -\frac{m_t}{t}. \tag{43}$$

Although m_t depends nonlinearly on retained information, it is a prescribed function of time along $y(t)$. Equation (43) is consequently a linear ODE for $z(t)$.

Its integrating factor is $\exp(-\int t^{-1}\,\mathrm{d}t) = 1/t$. Multiplication by this factor gives a product derivative, which can be integrated from 1 to t:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{z(t)}{t}\right) &= \frac{z'(t)}{t} - \frac{z(t)}{t^2} = -\frac{m_t}{t^2},\\
\frac{z(t)}{t} - z(1) &= -\int_1^t \frac{m_s}{s^2}\,\mathrm{d}s = \int_t^1 \frac{m_s}{s^2}\,\mathrm{d}s.
\end{aligned} \tag{44}$$

Multiplying by t yields the explicit solution

$$z(t) = tz(1) + t\int_t^1 \frac{m_s}{s^2}\,\mathrm{d}s. \tag{45}$$

We now evaluate the solution in Eq. (45) as $t \downarrow 0$. The initial-noise term $tz(1)$ vanishes, so it remains to determine the limit of the integral term. Assume that m_t is continuous along the principal trajectory and converges to $m_* = m_0(y(0), \ell)$ as $t \downarrow 0$. Separating this limiting value from the integral gives

$$t \int_t^1 \frac{m_s}{s^2} \, \mathrm{d}s = (1-t)m_* + t \int_t^1 \frac{m_s - m_*}{s^2} \, \mathrm{d}s. \tag{46}$$

The first term converges to m_*. To show that the remaining term vanishes, fix any $\eta > 0$. By continuity at $t = 0$, there exists $\delta \in (0, 1)$ such that $\|m_s - m_*\| < \eta/2$ whenever $0 < s < \delta$. Fix this δ and define

$$M_\delta = \int_\delta^1 \frac{\|m_s - m_*\|}{s^2} \, \mathrm{d}s < \infty, \tag{47}$$

which is finite by continuity on $[\delta, 1]$. For any $0 < t < \delta$, splitting the integral at δ gives

$$\begin{aligned} \left\| t \int_t^1 \frac{m_s - m_*}{s^2} \, \mathrm{d}s \right\| &\le \frac{\eta}{2} t \int_t^\delta \frac{1}{s^2} \, \mathrm{d}s + t \int_\delta^1 \frac{\|m_s - m_*\|}{s^2} \, \mathrm{d}s \\ &= \frac{\eta}{2} \left(1 - \frac{t}{\delta}\right) + t M_\delta \le \frac{\eta}{2} + t M_\delta. \end{aligned} \tag{48}$$

Now choose $0 < t < \min\{\delta, \eta/[2(M_\delta + 1)]\}$. The bound above is strictly smaller than η, proving that the remaining integral term converges to zero as $t \downarrow 0$. Since η was arbitrary, we obtain

$$\lim_{t \downarrow 0} t \int_t^1 \frac{m_s}{s^2} \, \mathrm{d}s = m_*, \qquad z(0) = m_0(y(0), \ell). \tag{49}$$

Restoring the nonzero mean gives

$$\begin{aligned} P_2 \widehat{X}(0) &= \mu_2 + m_0(y(0), \ell) \\ &= \mathbb{E}[P_2 x_0 \mid P_1 x_0 = \mu_1 + y(0), \ell] \\ &= \mathbb{E}[P_2 x_0 \mid P_1 x_0 = P_1 \widehat{X}(0), \ell]. \end{aligned} \tag{50}$$

The second line follows from $Y_0 = P_1 x_0 - \mu_1$, and the last uses $P_1 \widehat{X}(0) = \mu_1 + y(0)$. This proves Proposition 2: under the stated assumptions, the discarded generated coordinates equal their conditional means given the retained endpoint and class, even when these means depend nonlinearly on the retained information.

B.3 Comparison with Direct v-Prediction and Implications for RiT

We now examine what happens when a direct v-predictor is given the same low-rank observation (Y_t, t, ℓ). Since $Y_t = (1-t)Y_0 + tY_1$, the velocity along the retained subspace satisfies $Y_1 - Y_0 = (Y_t - Y_0)/t$. Taking the conditional expectation gives

$$\mathbb{E}[Y_1 - Y_0 \mid Y_t, t, \ell] = \frac{Y_t - \mathbb{E}[Y_0 \mid Y_t, t, \ell]}{t}. \tag{51}$$

This is exactly the retained velocity induced by the optimal x-predictor in the previous subsection. Thus, both objectives generate the same principal trajectory $y(t)$ under optimal regression, and we can compare their behavior in the discarded subspace along this trajectory. The target velocity of the discarded components is $P_2(x_1 - x_0) = Z_1 - Z_0 - \mu_2$. Its optimal MSE prediction is obtained by taking the conditional expectation:

$$\begin{aligned} \mathbb{E}[P_2(x_1 - x_0) \mid Y_t, t, \ell] &= \mathbb{E}[Z_1 \mid Y_t, t, \ell] - \mathbb{E}[Z_0 \mid Y_t, t, \ell] - \mu_2 \\ &= -m_t(Y_t, \ell) - \mu_2. \end{aligned} \tag{52}$$

Here $\mathbb{E}[Z_1 \mid Y_t, t, \ell] = 0$ because the discarded Gaussian noise is independent of the retained observation, as established above. Along the principal trajectory, the generated minor state $u(t) = (1-t)\mu_2 + z(t)$ therefore satisfies $u'(t) = -\mu_2 - m_t$. Since $u'(t) = -\mu_2 + z'(t)$, we obtain $z'(t) = -m_t$. Integrating backward from $t = 1$ gives

$$z(t) = z(1) + \int_t^1 m_s \, \mathrm{d}s, \qquad z(0) = z(1) + \int_0^1 m_s \, \mathrm{d}s. \tag{53}$$

The integral depends only on the principal trajectory and class label, while $z(1) = Z_1$ is independent Gaussian noise. Consequently, conditioning on $y(\cdot)$ and ℓ, the integral becomes fixed and the remaining randomness comes entirely from Z_1:

$$\mathrm{Cov}(z(0) \mid y(\cdot), \ell) = \mathrm{Cov}(Z_1 \mid y(\cdot), \ell) = P_2. \tag{54}$$

The direct v-predictor can shift the initial noise according to the retained information, but its covariance remains unchanged. Without access to the discarded state, it cannot generally reproduce the conditional distribution of the clean-data minor components.

The two prediction objectives therefore produce different results under the same low-rank observation. As proved above, x-prediction removes the initial minor noise and reconstructs the centered discarded coordinates as $m_0(y(0), \ell)$. This allows the backbone to focus on retained information while completing the missing components through their conditional means. However, the reconstructed minor components are deterministic given the retained endpoint and class, whereas the original data may contain additional conditional variation. Direct v-prediction instead preserves the discarded Gaussian noise with unit variance and cannot adapt its distribution without observing that noise.

These results motivate the two components of RiT. By directly receiving the full noisy state, the raw-conditioned nonlinear decoder can process information discarded by the backbone bottleneck, allowing the backbone to focus on retained information while the decoder models nonlinear dependencies and conditional variation in minor components. This provides greater expressive capacity than the conditional-mean completion of the restricted x-predictor. The output-side shortcut further provides a learnable approximation to the affine transport analyzed in Appendix A, reducing the need for the decoder to learn this simpler contribution through its nonlinear layers. Together, these components allow RiT to retain a compact backbone representation while modeling richer joint conditional distributions in pixel space.

B.4 Effect of the Time-denominator Cutoff

The preceding analysis shows that the discarded components converge to their conditional means under the unclipped x-prediction flow. In practice, the velocity denominator is clipped as $\bar{t} = \max(t, \epsilon)$ to improve numerical stability near $t = 0$. We now examine how this modification affects the endpoint behavior. The clipped and unclipped equations coincide for $t \geq \epsilon$. To isolate the final interval, we hold the principal trajectory fixed and let $M_t = \mu_2 + m_t$. The clipped minor state $u_\epsilon(t)$ starts from $u_\epsilon(\epsilon) = u(\epsilon)$ and satisfies

$$u'_\epsilon(t) - \frac{u_\epsilon(t)}{\epsilon} = -\frac{M_t}{\epsilon}, \qquad 0 \leq t \leq \epsilon. \tag{55}$$

Multiplication by $e^{-t/\epsilon}$ gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(e^{-t/\epsilon} u_\epsilon(t)\right) = -\frac{1}{\epsilon} e^{-t/\epsilon} M_t. \tag{56}$$

Integrating over $[0, \epsilon]$ yields

$$e^{-1} u(\epsilon) - u_\epsilon(0) = -\frac{1}{\epsilon} \int_0^\epsilon e^{-s/\epsilon} M_s \, \mathrm{d}s, \tag{57}$$

and consequently

$$u_\epsilon(0) = e^{-1} u(\epsilon) + \frac{1}{\epsilon} \int_0^\epsilon e^{-s/\epsilon} M_s \, \mathrm{d}s. \tag{58}$$

Because $\epsilon^{-1} \int_0^\epsilon e^{-s/\epsilon} \, \mathrm{d}s = 1 - e^{-1}$, subtracting $M_0 = \mu_2 + m_*$ gives

$$\begin{aligned} u_\epsilon(0) - M_0 &= e^{-1}\big(u(\epsilon) - M_0\big) + \frac{1}{\epsilon} \int_0^\epsilon e^{-s/\epsilon} (m_s - m_*) \, \mathrm{d}s, \\ \|u_\epsilon(0) - M_0\| &\leq e^{-1} \|u(\epsilon) - M_0\| + (1 - e^{-1}) \sup_{0 \leq s \leq \epsilon} \|m_s - m_*\|. \end{aligned} \tag{59}$$

The first term vanishes as $\epsilon \downarrow 0$ by the endpoint proof, and the second vanishes by continuity. Small clipping gives a nearby solution when the conditional mean varies regularly near the endpoint, with

its effect controlled by the two terms in this bound. To track the remaining noise, substitute $t = \epsilon$ into Eq. (45) and restore the mean:

$$u(\epsilon) = (1-\epsilon)\mu_2 + \epsilon Z_1 + \epsilon \int_\epsilon^1 \frac{m_s}{s^2}\,\mathrm{d}s. \tag{60}$$

Only the term ϵZ_1 depends on the discarded initial noise. Equation (58) multiplies it by e^{-1}, leaving $\epsilon e^{-1} Z_1$ at the endpoint. At $\epsilon = 0.05$, the residual noise standard deviation is approximately 0.0184 in each orthonormal minor coordinate, compared with one under restricted direct v-prediction. Thus, when the conditional mean varies smoothly near $t = 0$, the clipped solution remains concentrated near the unclipped conditional-mean endpoint, with only a small amount of residual noise. This analysis isolates the effect of the cutoff along a fixed principal trajectory, while numerical integration, regression error, and changes to the principal trajectory introduce additional approximation effects.

B.5 A Visual Illustration of Principal-component Reconstruction

We compare an image with its affine reconstruction using the leading 128 of 768 patch components in Figure 6. Although many coordinates are discarded, much of the visible structure remains. This illustrates how the retained components can provide context for completing missing details, making conditional-mean reconstruction a plausible use of a compact representation. The displayed reconstruction is a linear PCA projection rather than the conditional-mean decoder analyzed above; it does not establish that discarded details are uniquely determined by the retained components.



Figure 6: **Principal-component reconstruction.** Original image (left) and its reconstruction using the leading 128 of 768 principal components (right). Much of the visible structure is preserved, illustrating how retained principal components can provide sufficient information for conditional completion, although some fine details may remain ambiguous.

C Implementation Details

C.1 Architecture and Optimization

The backbones follow the B, L, and H configurations of JiT [30]. The principal architectural settings are listed in Table 6. The input embedding consists of two linear maps with the indicated intermediate width, which constrains the learned backbone input map without changing the pixel-space data or noise. We use fixed two-dimensional sinusoidal input position embeddings, two-dimensional rotary position embeddings in attention, RMS normalization, and SwiGLU feed-forward layers, with the same block type in the lightweight decoder. Thirty-two class tokens are inserted after 4, 8, and 10 backbone blocks for B, L, and H, respectively. These tokens participate in the remaining backbone blocks and are removed before the image features enter the decoder.

Training images are resized and center-cropped to 256×256, randomly flipped horizontally with probability 0.5, and normalized to $[-1, 1]$. We use AdamW with global batch size 1024, betas $(0.9, 0.95)$, and zero weight decay. The learning rate increases linearly to 2×10^{-4} over the first 6250 updates and remains constant thereafter for B and L. We sample $\mathrm{logit}(t) \sim \mathcal{N}(0.8, 0.8^2)$ unless explicitly varying the time distribution. The class condition is dropped with probability 0.1 to train the unconditional branch used by CFG, while attention and projection dropout are both zero. RiT-B

Setting	RiT-B/16	RiT-L/16	RiT-H/16
Image resolution	256×256	256×256	256×256
Backbone patch size	16	16	16
Backbone depth	12	24	32
Backbone hidden width	768	1024	1280
Backbone attention heads	12	16	16
Input bottleneck width	128	128	256
In-context class tokens	32	32	32
Blocks before class-token insertion	4	8	10
Decoder patch size	8	8	8
Decoder depth	2	2	2
Decoder hidden width	384	512	768
Decoder attention heads	6	8	12
Training updates	750K	750K	600K
Training epochs	600	600	480

Table 6: **Final model configurations.** All final models use full pixel-space data and noise, a raw-conditioned decoder, and the shared-denominator anchored shortcut. Epoch counts are approximate. PCA projection is used only in the diagnostic experiments, rather than in the final models.

is trained on 8 NVIDIA A800 GPUs with 80 GB of memory per GPU. RiT-L and RiT-H each use 16 NVIDIA A800 GPUs with the same memory capacity, while retaining the global batch size of 1024.

For RiT-H, we multiply the effective learning rate of the raw shortcut input/output projections and the parameters producing c_1, c_2, c_3 by 0.01 after 375K updates, corresponding to approximately 300 epochs. The remaining parameters, including the semantic projection and the parameters producing c_4, c_5, retain the base learning rate. This is a scale-specific stabilization choice rather than a change to the architecture or training objective, and it is not used for the 250K ablations or the B/L runs. We maintain EMA models with decay rates 0.9996, 0.9998, and 0.9999 and compare their sampling performance. The reported configurations in Table 8 use 0.9996, selected empirically from the evaluated EMA settings.

The backbone and decoder residual modulation layers are zero-initialized following the baseline convention. For shortcut variants, the final semantic projection and coefficient-prediction outputs are also zero-initialized. The raw shortcut input projection is initialized with orthonormal columns, and its output projection is initialized with the transposed weight. Both projections have zero-initialized biases. The input projection used to inject raw patches into the decoder blocks is initialized separately and is not shared with the shortcut. These are initialization choices only, and the projection weights remain learnable without orthogonality constraints or transpose tying during training.

C.2 Shortcut Variants and Controlled Comparisons

Table 7 specifies the coefficient assignments of the output-shortcut variants. The decoder condition c is the sum of time and class embeddings, projected to the decoder width. One SiLU–linear mapping produces four, five, five, or six channel vectors for the basic, free fractional, anchored fractional, or separate-denominator variant, respectively. The names c_i refer to the ordered outputs within each variant, following the implementation. The symbols γ and β denote the corresponding coefficients c_i in Table 7 and do not introduce additional modulation parameters.

For these shortcut variants, the branch features are

$$r = X_{\text{in}} x_t^{(s_d)}, \qquad q = H_{\text{out}}(\text{RMSNorm}(h) \odot (1+\gamma) + \beta)\,, \tag{61}$$

and their common pixel-patch output is $X_{\text{out}}(g_r \odot r + g_h \odot q)$. The semantic projection H_{out} acts before g_h, and the two branches are added before the shared output projection. For brevity, we define

$$\begin{aligned} n_i &= 1 + (1-t)c_i, \qquad d_i = \max\{t + (1-t)c_i, \epsilon\}, \\ f_i &= \max\{t + c_i, \epsilon\}, \qquad \epsilon = 0.05. \end{aligned} \tag{62}$$

These quantities are channel-wise and shared across all decoder tokens, depending only on the time and class condition. The raw and semantic features r and q are computed for each token from the

noisy input and decoder hidden states, respectively. The two raw-input projections X_{proj} and X_{in} remain distinct, as described in Section 4.5.

Variant	g_r	g_h	(γ, β)
Basic	$1 + c_1$	$1 + c_2$	(c_3, c_4)
Free fractional	$(1 + c_1)/f_2$	$(1 + c_3)/f_2$	(c_4, c_5)
Anchored fractional	n_1/d_2	n_3/d_2	(c_4, c_5)
Anchored, separate denominators	n_1/d_2	n_3/d_4	(c_5, c_6)

Table 7: **Coefficient assignments for the output-shortcut ablation.** All operations are channel-wise. The final RiT uses the anchored fractional variant with a shared denominator, while retaining separate learned numerators for the raw and semantic branches.

The no-shortcut variant instead produces two coefficients and directly predicts the pixel patch as $H_{\text{out}}(\text{RMSNorm}(h) \odot (1 + c_1) + c_2)$. In this variant, H_{out} maps to the pixel-patch dimension rather than the decoder hidden width, and there is no additional X_{in} or X_{out} branch. It can still receive raw input inside the decoder through X_{proj}. Thus, disabling decoder injection and removing the output shortcut are separate interventions. In Table 2, all decoder rows remove the output shortcut, and the width-384 "No" row additionally removes raw-input injection into the decoder blocks.

With the initialization described above, the no-shortcut readout initially outputs zero. For the tested shortcut models, the raw input/output composition is initially the identity, so the basic shortcut initially outputs x_t, whereas the fractional variants initially output $x_t/\max(t, \epsilon)$. These initial outputs are determined by the architecture and initialization, independently of the prediction objective. During training, the network output is supervised as either velocity or clean data, depending on whether v-prediction or x-prediction is used.

PCA experiments use $k \in \{128, 192, 256, 384, 768\}$, backbone width 768, embedding intermediate width 768, no added decoder blocks, and no output shortcut. Data and noise are projected by the same affine map in Eq. (5). The vanilla-L intervention has hidden and embedding intermediate widths 1024, 24 blocks, and patch size 16. The vanilla patch-8 intervention keeps a 12-block width-768 backbone but reduces raw patch dimension to 192. Neither intervention imposes a low-rank input bottleneck. All these controls use the same global batch size of 1024 and are trained for 250K updates, allowing us to examine the effects of backbone capacity and raw patch dimensionality under the same training schedule. Their computational costs differ, since increasing backbone width or reducing patch size changes the amount of computation required per update.

For the decoder and shortcut ablations, the B backbone has a width-128 input bottleneck. Decoder widths 128, 256, and 384 use 2, 4, and 6 attention heads, respectively. The final B model is the shared-denominator anchored setting. Bottleneck and time-sampling studies vary only their named setting relative to this model. In the corresponding x-prediction shortcut experiment, the architecture and initialization are unchanged, but its output is trained as a clean-image prediction using Eq. (4) and converted to velocity during sampling. These comparisons use a common training recipe to isolate the effects of the tested architectural and training choices, without additional hyperparameter tuning for individual variants.

C.3 Sampling and Evaluation

We use 50 uniformly spaced integration steps from $t = 1$ to $t = 0$. The first 49 steps use the second-order Heun update, and the final step uses a single Euler update directly to the data endpoint. This avoids evaluating the velocity at $t = 0$, where the unregularized x-prediction parameterization contains a singular $1/t$ factor. Our implementation also clips this denominator as described in the preliminary section; we retain the same final-step convention for both objectives. The resulting sampler requires 99 velocity-function evaluations over 50 integration steps. CFG evaluates conditional and unconditional predictions together and combines them as

$$v_{\text{CFG}} = v_{\varnothing} + w\,(v_{\ell} - v_{\varnothing}). \tag{63}$$

Guidance is active only within the specified time interval, and we use $w = 1$ outside that interval. Each guided evaluation computes both conditional and unconditional predictions in a doubled batch,

Model	Updates	Epochs	EMA	CFG scale	CFG interval
RiT-B/16	250K	200	0.9996	3.1	$[0.0, 0.89]$
RiT-L/16	250K	200	0.9996	2.5	$[0.0, 0.89]$
RiT-H/16	250K	200	0.9996	2.4	$[0.0, 0.89]$
RiT-B/16	750K	600	0.9996	2.8	$[0.0, 0.89]$
RiT-L/16	750K	600	0.9996	2.6	$[0.0, 0.89]$
RiT-H/16	600K	480	0.9996	2.35	$[0.0, 0.90]$

Table 8: **Inference configurations for the reported guided results.** Epoch counts are approximated from training updates and batch size. All configurations use 50-step Heun sampling with a final Euler step. Unguided ablations use $w = 1$ and EMA 0.9996 with the same integration procedure.

requiring approximately twice the computation of an unguided evaluation. The reported 99 velocity-function evaluations count these combined predictions as one solver evaluation.

FID-50K and IS are computed on 50,000 class-conditional samples using the ADM evaluation implementation [11] and its ImageNet reference statistics. Generated images are mapped from $[-1, 1]$ to 8-bit RGB by rounding and clipping. We select guidance settings separately for different model scales and training horizons. Table 8 summarizes the configurations used to obtain the reported results, while unguided ablations are evaluated with CFG disabled.

C.4 Patch Statistics and Representation Probes

The patch statistics are computed over the ImageNet training set after the same resize, crop, and intensity normalization as training. The empirical covariance is centered by the patch mean. We account for horizontal flips with probability 0.5, including the between-mean covariance term when combining original and flipped patch distributions. These statistics define the affine PCA intervention and diagnostic plots, but are not required by the final RiT models during either training or sampling.

The embedding analysis uses the ordinary training weights, not EMA, at the recorded checkpoints. We compose the two consecutive linear embedding weights as W_2W_1 and compute the effective rank from the singular-value entropy in double precision. The PCA energy ratio uses a fixed $K = 280$ for every model and checkpoint, without eigenvalue weighting, so it measures the learned map's response to the same set of unit-norm PCA directions. The full-L curve corresponds to the vanilla-L control. In particular, the PCA-128 curves describe training with projected data and noise while retaining a full-width embedding. They should therefore be distinguished from the architectural input-bottleneck ablation, which keeps the full pixel-space distributions unchanged.

The reconstruction probes use the ordinary 250K weights in evaluation mode and a fixed set of 1024 training images, processed in 16 batches of 64. We construct noisy inputs using the same interpolation and time-sampling procedure as training, with Gaussian noise $x_1 \sim \mathcal{N}(0, I)$ and $\mathrm{logit}(t) \sim \mathcal{N}(\mu_t, \sigma_t^2)$. The parameters (μ_t, σ_t) follow the corresponding training configuration, and the noisy state is formed as $x_t = (1 - t)x_0 + tx_1$. Image selection, horizontal flips, Gaussian noise draws, and the underlying standard-normal draws for the time samples are fixed across the compared models. For PCA runs, the same affine projection is applied to data and noise before forming the noisy input. The forward passes use full precision and retain the class labels without CFG dropout.

We collect image-token features after each backbone block, excluding the in-context class tokens. For each layer, we concatenate the features and corresponding noisy patches from all 16 batches, then fit a single affine least-squares regression in double precision. We compute the R^2 score on the same data used to fit the regression. This score measures how much of the noisy input x_t can be linearly reconstructed from the hidden states at each layer.

D Limitations and Broader Impact

Limitations. Our experiments focus on class-conditional ImageNet generation at 256×256 and B, L, and H model scales. Generalization to higher resolutions, text-conditioned generation, other data distributions, and larger models remains to be studied. RiT provides a simple and effective realization

of the identified bandwidth principle, while more expressive and efficient raw-conditioned decoder designs may offer further improvements in future work.

Broader Impact. Our work provides insights into the relationship between prediction objectives and architectural information flow, which may facilitate the development of more effective end-to-end pixel-space generative models. As with other image generation methods, improved synthesis capabilities may also be misused to produce misleading content or deepfakes. Since this work focuses on fundamental modeling principles, we do not introduce additional deployment-specific safeguards, which remain important considerations for practical applications.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have clearly introduced our contribution to pixel-space Flow Matching in our abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations of our work in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We state the assumptions and main results in Section 4, with complete derivations and proofs provided in Appendices A and B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have discussed our architecture and used hyper-parameters in our method and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The paper provides a link to our public code repository, which includes training and evaluation instructions and guidance for preparing the dataset.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided these details in our method and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to the high computational cost of training generative models on the ImageNet dataset, it is not common practice to report error bars from multiple training runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the training configurations and computational resources in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed the broader impacts in the appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work focuses on foundational research in pixel-space generative modeling rather than releasing a high-risk pretrained model or dataset. Potential misuse is discussed in the broader impact section.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original datasets and code used in our experiments and comply with their applicable licenses and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our released code is documented in the public repository linked in the paper, including implementation details and usage instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our work does not involve crowdsourcing.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our work does not involve human-subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as important, original, or non-standard components of the core methodology. Any assistance was ancillary to the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.