BEYOND THE ESTIMATE: WIENER-INFORMED DECODING FOR EASIER TRAINING OF PIXEL-SPACE FLOW MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning velocity fields directly in pixel space is difficult and slow. We show that coarse second-order image statistics can ease this task: frozen Wiener estimates improve velocity learning, with gains extending beyond the linear predictions they explicitly provide. Moreover, we observe similar but complementary gains from frozen bases and pixel decoding. These observations motivate **W**iener-**i**nformed **D**ecoding (WiD), which uses statistical knowledge both for direct prediction and to organize decoder computation without modifying native blocks. WiD combines a *trainable Wiener prediction path with coefficient interfaces* that inject features into the native decoder and return its predicted coefficient corrections, with these components initialized with data statistics. All introduced parameters, including the spectrum and coordinate transforms, are trained jointly under the original pixel-space velocity objective. Across model scales and decoder architectures, WiD accelerates training and improves generation quality with modest parameter overhead. It also yields smoother gradient trajectories and remains effective with representation alignment. On ImageNet 256×256, PixelDiT-XL with WiD reaches FID 1.61 at 260 epochs, matching the baseline's reported quality at 320 epochs with less than 1% additional parameters. We also validate the early gains on text-to-image tasks. These results show that statistical denoisers can contribute beyond the estimate: the computational structure also helps pixel generation learning.

1 INTRODUCTION

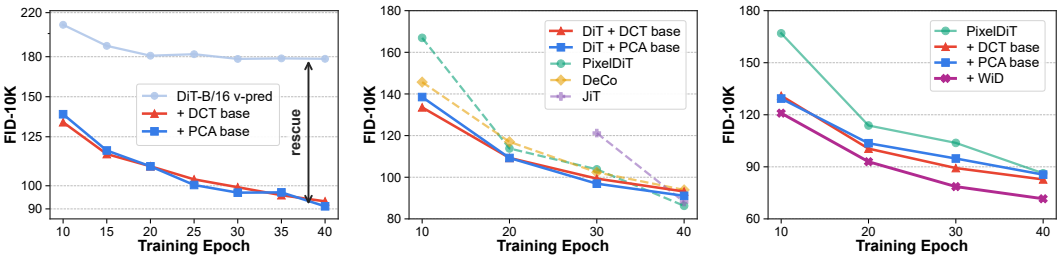
Pixel-space flow models offer a simpler generative pipeline, but not an easier learning problem. By operating directly in pixel space, they remove the need for a separate, pretrained autoencoder. However, the denoising network must operate on noisy high-dimensional pixels while learning natural image structure, which makes optimization slow and challenging (Li & He, 2026; Ma et al., 2026).

Recent work addresses the challenge in two prominent directions. (i) Prediction-space approaches seek more learnable target parameterizations, such as clean-image prediction (Li & He, 2026) and rank-asymmetric velocity prediction (Chen et al., 2026a). (ii) Architectural approaches distribute contextual modeling and dense pixel prediction across specialized components. For example, DeCo (Ma et al., 2026) and PixelDiT (Yu et al., 2026) pair a contextual backbone with a fine-grained pixel decoder. These advances make pixel generation more tractable by changing the parameterization or computational organization—and motivate a closer look at the pixel denoising task itself.

However, high dimensionality does not imply an absence of exploitable structure. Natural images often possess correlations across space and color, whereas the added Gaussian noise is isotropic. The second-order statistics yield a closed-form linear denoiser—*the Wiener estimator* (Wiener, 1949; Kailath et al., 2000). Although this estimator cannot capture the full complexity of natural images, it helps previous studies (Wang & Vastola, 2024; Lukoianov et al., 2025) investigate the learned denoising behavior and locality. Motivated by these observations, we ask a training question: *is it possible to ease pixel-space velocity learning by explicitly providing this statistical estimate?*

Observation: a fixed estimate already makes a substantial difference. We construct a fixed Wiener estimator from statistics to provide an analytic velocity base and train the DiT (Peebles & Xie, 2023) to predict the residual correction under the velocity objective. We consider two typical construc-

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



(a) Analytic bases rescue direct velocity prediction DiT in pixel space.

(b) Analytic bases yield competitive quality under native recipes.

(c) WiD provides stronger and more persistent gains on PixelDiT.

Figure 1: **Analytic Wiener bases ease pixel-space velocity learning.** We train B-size models for 40 epochs on ImageNet 256×256 w/o REPA and report FID-10K results w/o CFG. **(a)** With the same DiT-B/16 setup, adding a DCT or PCA Wiener base rescues the velocity prediction baseline. **(b)** DiTs with analytic bases reach the performance range of several pixel-space methods. Each method uses its native training and evaluation recipe. **(c)** When added to native PixelDiT, our Wiener-informed Decoding provides stronger and more sustainable gains than frozen Wiener bases.

tions: global DCT (Ahmed et al., 1974) for image-wide spectral structure and patch PCA (Zhang et al., 2010) for local spatial feature. Both analytic bases rescue the otherwise struggling plain DiT (Figure 1a), bringing generation quality into the range of contemporary approaches (Figure 1b). Consequently, even coarse statistical knowledge can ease the burden of velocity learning.

A frozen statistical prediction is not the whole story. For DiT, our diagnostics further show that much of the reduction in velocity prediction error lies beyond the fitted linear component (Figure 2a). Thus, the effect of Wiener bases is not confined to correcting a linear prediction. This leads us to consider the computation behind the estimate: the Wiener estimator transforms a noisy observation into statistical coordinates, adjusts its coefficients according to the noise level, and maps them back to pixels. *Can we go a step further and use these operations to organize the denoiser's computation, rather than only adding the resulting estimate to its output?*

Pixel decoding provides a natural and task-aligned interface for this integration. In PixelDiT and DeCo, the backbone provides semantic context, while the decoder maps noisy pixels to velocity—*the same input–output task* addressed by the Wiener estimator. We observe similar but complementary gains from frozen Wiener bases and pixel decoding (Figure 2b). Our input-perturbation tests support the task alignment: the direct pixel pathway shows closer Wiener alignment than the semantic pathway (Figure 2c). Therefore, we incorporate the estimator's analysis and synthesis operations into the pixel decoder as trainable components, while retaining the semantic backbone architecture.

Approach: from frozen bases to adaptive decoding. We introduce **Wiener-informed Decoding (WiD)** that incorporates statistical knowledge in two ways. (i) *A statistical starting point.* Using patch PCA statistics, we initialize a direct prediction path to produce the Wiener velocity estimate. It turns noisy patches into statistical coordinates, scales their coefficients according to the noise level, and maps them back to pixels. (ii) *Coordinates for decoder computation.* We also allow noisy-patch coordinates to integrate with the decoder's native input embeddings as additional features. Alongside its original pixel-space velocity output, the decoder also predicts a correction to the coefficient-space estimate. The sum of the Wiener-branch and pixel-head predictions is supervised by the original training objective, and introduced parameters are trained jointly with the network.

WiD improves training stability and efficiency, and accommodates different pixel-decoder architectures with a modest parameter increase. Empirically, it reaches comparable generation quality in fewer training steps, with gains that persist further into training than frozen bases (Figure 1c). On ImageNet 256 × 256, our PixelDiT-XL+WiD reaches an FID of 1.61 after 260 training epochs.

Contribution Summary. We summarize our contributions as follows.

- **Observation.** We show that coarse statistical estimates can ease pixel-space velocity learning, with error reductions beyond the fitted linear responses. The observation motivates us to use the computation behind the estimate, rather than only its output.

- **Method.** We introduce Wiener-informed Decoding (WiD), which integrates a trainable statistical prediction path with coefficient-based feature processing and output correction.
- **Results.** We demonstrate improved training stability, efficiency and generation-quality gains across different pixel-decoder architectures with modest overhead.

2 PRELIMINARIES

2.1 LINEAR WIENER ESTIMATION FOR FLOW MATCHING

Consider two independent Gaussian vectors $x \sim \mathcal{N}(\mu_x, C_x)$ and $\epsilon \sim \mathcal{N}(\mu_\epsilon, C_\epsilon)$ in $\mathbb{R}^d$, observed through a linear combination $z = \alpha x + \beta\epsilon$ with known scalar coefficients. The observation has mean $\mu_z = \alpha\mu_x + \beta\mu_\epsilon$ and covariance $C_z = \alpha^2 C_x + \beta^2 C_\epsilon$.

Assuming C_z is invertible, their Wiener estimates are

$$\mathbb{E}[x \mid z] = \mu_x + \alpha C_x C_z^{-1}(z - \mu_z), \qquad \mathbb{E}[\epsilon \mid z] = \mu_\epsilon + \beta C_\epsilon C_z^{-1}(z - \mu_z). \tag{1}$$

Under this Gaussian model, these conditional means minimize the expected squared estimation error and are affine functions of the observed mixture (Kailath et al., 2000).

For linear flow matching, we use standard Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$ and the interpolation

$$z_t = tx + (1-t)\epsilon, \qquad v = x - \epsilon, \tag{2}$$

where $t \in [0, 1]$ runs from noise to data and I is the identity matrix (Liu et al., 2022; Lipman et al., 2023; Albergo et al., 2025). Approximating the data distribution by $x \sim \mathcal{N}(\mu, C)$, define

$$A_t(C) = \big[tC - (1-t)I\big]\big[t^2 C + (1-t)^2 I\big]^{-1}. \tag{3}$$

Subtracting the two conditional means in equation 1 gives the analytic velocity estimate

$$\begin{aligned} V_W(z_t, t) &:= \mathbb{E}[x - \epsilon \mid z_t, t] \\ &= \mu + A_t(C)(z_t - t\mu). \end{aligned} \tag{4}$$

This expression is exact under the assumed Gaussian prior; for natural images, it provides a second-order statistical approximation to the conditional velocity.

In practice, we use a linear estimator *without an additive offset*. For this estimator class, the relevant statistic is the uncentered second moment $M = \mathbb{E}[xx^\top] = C + \mu\mu^\top$. With exact second moments, $A_t(M)z_t$ minimizes the expected squared velocity error among linear estimators without an additive offset (Kailath et al., 2000). Using a structured empirical approximation $\widetilde{M}$, we obtain

$$V_W(z_t, t) = A_t(\widetilde{M})z_t. \tag{5}$$

This formulation requires no explicit mean correction, while retaining the mean's contribution to the second moment. We explain how to construct such moments and use them in training in Section 3.

2.2 PIXEL DECODER INTERFACES

Architectures such as PixelDiT and DeCo separate contextual modeling from dense pixel prediction. Given a conditioning input y, their common interface can be written as

$$s = \mathcal{B}(z_t, t, y), \qquad \hat{v} = \mathcal{D}(z_t, s), \tag{6}$$

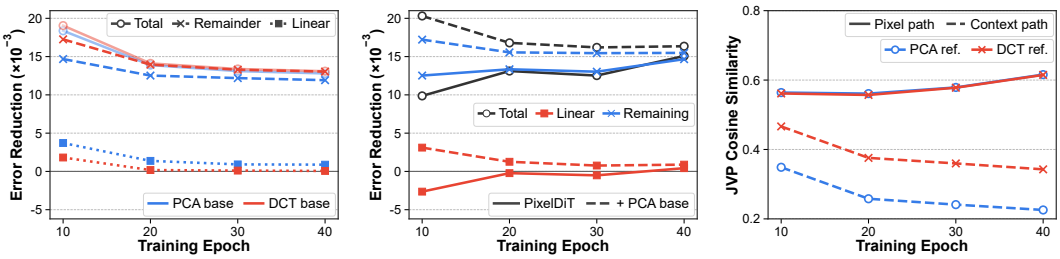
where $\mathcal{B}$ is a contextual backbone, s denotes the conditioning features, and $\mathcal{D}$ is the pixel decoder. Although the internal blocks may differ, the interface remains consistent. The decoder embeds noisy pixels, processes features under contextual modulation, and maps the features to a velocity prediction. We explain how to incorporate Wiener-informed operations into this interface in Section 4.

3 LEARNING WITH ANALYTIC WIENER BASES

3.1 STRUCTURED STATISTICAL REFERENCES

Let $\widetilde{M} \in \mathbb{R}^{d\times d}$ have the eigendecomposition $\widetilde{M} = U\Lambda U^\top$, with $U^\top U = I$ and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$, so that equation 5 becomes

$$V_W(z_t, t) = U\,\mathrm{diag}\big(a_j(t)\big)U^\top z_t, \qquad a_j(t) = \frac{t\lambda_j - (1-t)}{t^2\lambda_j + (1-t)^2}. \tag{7}$$



(a) Frozen bases improve beyond the fitted linear responses.

(b) A frozen base provides additional gains with pixel decoding.

(c) The pixel pathway shows greater alignment to Wiener bases.

Figure 2: **Prediction gains and pathway alignment. (a, b)** Using a well-trained official PixelDiT-XL checkpoint as the prediction reference, we fit affine maps to each model's predictions and decompose the prediction error into linear and residual components. Relative to plain DiT, we compare how Wiener bases and pixel decoding reduce MSE in each component: **(a)** DiT with frozen PCA or DCT bases; **(b)** PixelDiT and PixelDiT+PCA base evaluated with patch-affine fits. **(c)** PixelDiT JVPs compared with Wiener responses, removing components parallel to perturbation.

Estimating the statistics. We estimate uncentered second moments from $51{,}200$ randomly sampled ImageNet training images, using the same RGB center crop to 256×256 and $[-1, 1]$ scaling as training preprocessing settings, without class labels or random augmentation.

Two structured bases. For 256^2 RGB images, the full moment matrix has $(3 \times 256^2)^2$ entries. We approximate it in two practical ways. (i) *Global DCT* retains a 3×3 RGB moment at each frequency of the orthonormal 2-D DCT-II, omitting cross-frequency coupling. The block eigendecompositions, combined with the DCT, specify U and Λ. (ii) *Shared patch PCA* pools all non-overlapping 16^2 RGB patches across images and positions, matching the patch size of mainstream pixel-space flow models, and diagonalizes their shared 768×768 moment. The resulting uncentered PCA basis and spectrum are reused for every patch, without modeling cross-patch moments. Each construction stores 768^2 moment entries ($256^2 \times 3^2 = 768^2$ for DCT), rather than the full image matrix.

Residual learning. We freeze each base and train the model to predict the residual under the original velocity objective:

$$\hat{v}_\theta(z_t, t, y) = V_W(z_t, t) + r_\theta(z_t, t, y),$$
$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{x,y,\epsilon,t}\left[\frac{1}{d}\left\|\hat{v}_\theta(z_t, t, y) - (x - \epsilon)\right\|_2^2\right], \tag{8}$$

where y is the conditioning input. The network still receives the unmodified z_t and no trainable parameters are added. We use the same summed velocity field for sampling and evaluation.

3.2 Beyond the Additive Linear Prediction

An analytic base supplies a prediction that the network would otherwise have to learn. But does it also help the network learn the remaining prediction? We investigate this by comparing the velocity predictions with a well-trained reference model: the official PixelDiT-XL checkpoint at 320 epoch.

At each fixed t, we fit an affine prediction ℓ_f from z_t to each model's complete velocity output f, and similarly fit ℓ_{ref} to the reference f_{ref}. For PCA, the fit is a shared patch-affine map; for DCT, it includes independent RGB-affine maps at each frequency. Each affine family contains the corresponding frozen Wiener base as a zero-bias special case, and subtracting the fitted response removes its explicit additive contribution. We measure errors in decomposed parts:

$$E_{\mathrm{lin}}(f) = \frac{1}{d}\mathbb{E}\|\ell_f - \ell_{\mathrm{ref}}\|^2, \qquad E_{\mathrm{rem}}(f) = \frac{1}{d}\mathbb{E}\|(f - \ell_f) - (f_{\mathrm{ref}} - \ell_{\mathrm{ref}})\|^2. \tag{9}$$

We fit by least squares on $4{,}096$ validation images and score on $3{,}072$ disjoint images with two fixed noise draws each. Figure 2a reports the reduction in each error while applying Wiener bases to DiT, averaged over the five time points. We observe that the remaining prediction contributes most of the improvement, and that the linear-error reduction shrinks with training. These results suggest that a frozen base does more than provide a better starting prediction: it may also help the network organize its computation to learn the remaining prediction more effectively.

3.3 Pixel Decoding and Statistical Prediction

Pixel decoding organizes velocity learning by separating contextual modeling from dense pixel prediction. We examine its effect using the same reference and patch-affine decomposition as above. Relative to plain DiT, native PixelDiT's improvement also lies primarily in the remaining prediction (Figure 2b). However, these gains do not make the statistical estimate redundant. Adding a frozen PCA base to PixelDiT further reduces both the fitted linear error and the remaining error. These additional gains generally narrow over training, paralleling the trend in FID gains in Figure 1c.

We next compare the responses of native PixelDiT's two input pathways with those of the Wiener estimator. With t and y fixed, we compute each pathway's Jacobian–vector product (JVP) for the same isotropic unit perturbation δ to z_t. The interface in equation 6 gives

$$j_{\text{pix}} = (\partial_{z_t}\mathcal{D})\delta, \qquad j_{\text{ctx}} = (\partial_s\mathcal{D})(\partial_{z_t}\mathcal{B})\delta. \tag{10}$$

The first holds the contextual features s fixed; the second holds the decoder's direct pixel input fixed. We compare these output-velocity JVPs with the Wiener responses $A_t(\widetilde{M})\delta$, after removing each response's component parallel to δ. Cosine similarities are averaged over 256 held-out images and $t \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$. On average, the direct pixel pathway is more aligned with both statistical references than the contextual pathway, and the gap widens over training (Figure 2c).

Together, these observations motivate us to integrate the statistical estimator into pixel decoding. (i) Frozen bases and native pixel decoding both improve the remaining prediction, yet their combination yields further gains, showing similar but also complementary benefits. (ii) The direct pixel pathway's closer alignment with Wiener responses supports the decoder as a natural interface for this integration. To pursue more persistent gains, we make the estimator's prediction path and coordinate transforms trainable parts of the decoder, allowing them to evolve with learning (Section 4).

4 Wiener-informed Decoding

In this section, we describe how to integrate the statistical estimator into pixel decoding as a trainable component. As illustrated in Figure 3, we propose Wiener-informed Decoding (WiD), which couples a trainable Wiener prediction path with coefficient feature processing and output correction.

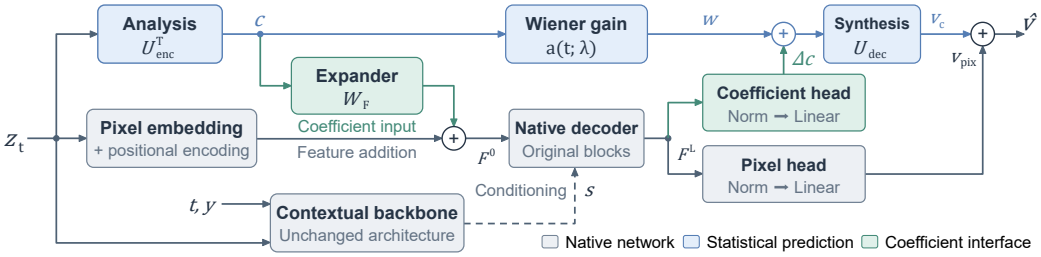


Figure 3: **Overview of our WiD framework.** WiD integrates statistical operations as trainable components with the native decoder, and allows both branches to interact through coefficient interfaces.

4.1 Architecture

Wiener path. We choose the full shared patch PCA basis for implementation, because it works in the same patch-level spatial space as the pixel decoder. For a noisy RGB patch $z_{t,i} \in \mathbb{R}^m$, where $m = 3p^2$, an analysis transform $U_{\text{enc}} \in \mathbb{R}^{m\times m}$ extracts coefficients and applies the Wiener gain:

$$c_i = U_{\text{enc}}^{\top} z_{t,i}, \qquad w_i = a(t;\lambda) \odot c_i. \tag{11}$$

The statistical prediction is obtained by mapping w_i back to pixels through a synthesis transform $U_{\text{dec}} \in \mathbb{R}^{m\times m}$. Here U_{enc}, U_{dec}, and λ are trainable parameters, and $a(t;\lambda)$ follows equation 7.

Coefficient interaction. The same coefficients, *before* Wiener scaling, also enter the native decoder. A linear expander $W_F \in \mathbb{R}^{p^2h\times m}$ maps them to pixel-level features of width h, which are added to

the native pixel embedding $\mathcal{E}$:

$$F_i^0 = \mathcal{E}(z_{t,i}) + \text{reshape}(W_F c_i), \qquad F^L = \mathcal{T}(F^0, s). \tag{12}$$

Here $\mathcal{T}$ denotes the native decoder blocks operating on features and contextual condition s. The decoder now predicts both a residual pixel-space velocity and a coefficient-space correction Δc:

$$\begin{aligned} \Delta c_i &= W_\Delta \,\text{vec}\big(\mathcal{N}_\Delta(F_i^L)\big) + b_\Delta, \\ \hat{v}_i &= \underbrace{U_{\text{dec}}\big[w_i + \Delta c_i\big]}_{\text{Wiener-branch prediction}} + \underbrace{\mathcal{H}_{\text{pix}}(F^L)_i}_{\text{native pixel prediction}} . \end{aligned} \tag{13}$$

The normalization $\mathcal{N}_\Delta$ follows the native output head's granularity. And the coefficient head maps the normalized features to m-dimensional correction vectors with linear projection W_Δ and bias b_Δ.

Thus, the statistical coordinates participate in both feature processing and velocity prediction, rather than only supplying an additive answer. We assemble the patch predictions into the image and supervise the sum $\hat{v}_\theta$ of both predictions with the original velocity objective in equation 8. Auxiliary objectives such as REPA (Yu et al., 2025), when used, remain those of the underlying training recipe.

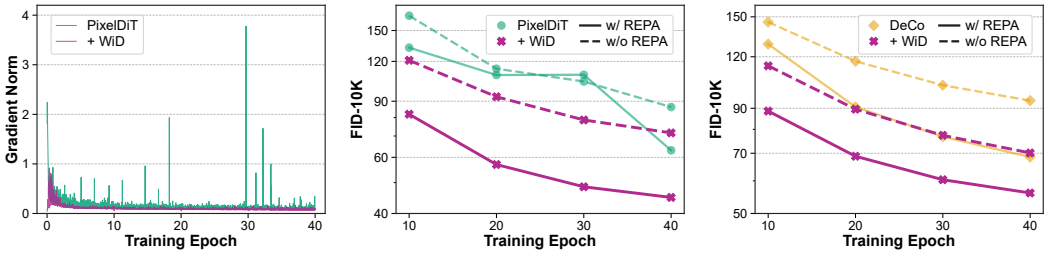
4.2 Initialization

Let U_0 and λ_0 denote the shared patch PCA basis and spectrum estimated in Section 3.1. We initialize U_{enc} and U_{dec} with U_0, and parameterize the spectrum in log space, initialized to $\log \lambda_0$. The coefficient correction and native pixel-output projections are both zero-initialized. Therefore, the initial total prediction $\hat{v}_\theta$ equals the patch PCA Wiener estimate V_W as a starting point.

At the input of native decoder, we split the content between the two routes. Let E_0 be the original per-pixel RGB projection and $\mathcal{L}(E_0)$ its patchwise lift, mapping a patch to flattened pixel features. We initialize the pixel projector $E = 0.5E_0$ and our expander $W_F = 0.5\mathcal{L}(E_0)U_0$. Thus, at initialization, the contributions of two routes are balanced, and the combined embedding satisfies

$$\mathcal{L}(E)z_{t,i} + W_F U_{enc}^\top z_{t,i} = \mathcal{L}(E_0)z_{t,i}. \tag{14}$$

Biases and positional contributions remain single-copy in the native embedding, so the complete initial features are also preserved. No special initialization is imposed on the internal decoder blocks. WiD thus starts from the Wiener prediction while preserving the native decoder's initial input features and block initialization. All introduced parameters are trained jointly with the network. The analysis and synthesis transforms evolve independently, without orthogonality or inverse constraints.



(a) WiD contributes to smoother gradient trajectories on PixelDiT-B.

(b) WiD consistently improves generation quality w/ and w/o REPA.

(c) WiD is also effective in improving the generation quality of DeCo.

Figure 4: **WiD improves training stability, remains compatible with REPA, and applies across decoder architectures.** We compare B-size models trained for 40 epochs on ImageNet 256×256. **(a)** PixelDiT+WiD exhibits smoother pre-clipping gradient norms than native PixelDiT with REPA. **(b,c)** WiD improves both PixelDiT and DeCo and gains persist with REPA.

4.3 Training Observations

Improved training stability. We report gradient norms before clipping under the same training recipe of PixelDiT-B with and without WiD in Figure 4a, both using REPA. Applying WiD leads to smoother gradient trajectories and fewer large excursions than the native model.

Compatibility with REPA. In the four-way PixelDiT-B comparison in Figure 4b, combining WiD with REPA yields further gains in generation quality, supporting their complementary use.

Generality across decoders. We also test WiD on DeCo-B with a different design of decoder architecture from PixelDiT (Figure 4c). Under DeCo's training recipe with its auxiliary frequency-domain objective, WiD consistently improves the generation quality over the native DeCo baseline.

Table 1: Class-conditional generation on ImageNet 256 × 256. Gray entries denote latent-space models. ↑ and ↓ denote higher and lower values are better, respectively. Our Wiener-informed Decoding consistently improves the generation quality of PixelDiT-XL during the training process.

Method	Epochs	#Params.	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
REPA-XL (Yu et al., 2025)	800	675M	1.42	4.70	305.7	0.80	0.65
DDT-XL (Wang et al., 2026b)	400	675M	1.26	–	310.6	0.79	0.65
RAE-XL (Zheng et al., 2026)	800	839M	1.13	–	262.6	0.78	0.67
ADM (Dhariwal & Nichol, 2021)	400	554M	4.59	5.25	186.7	0.82	0.52
RIN (Jabri et al., 2023)	480	410M	3.42	–	182.0	–	–
VDM++ (Kingma & Gao, 2023)	–	–	2.12	–	267.7	–	–
RDM (Teng et al., 2024)	400	553M + 553M	1.99	3.99	260.4	0.81	0.58
PixelFlow-XL (Chen et al., 2025b)	320	677M	1.98	5.83	282.1	0.81	0.60
EPG-XXL/16 (Lei et al., 2026)	600	789M	1.81	–	294.6	0.80	0.61
PixNerd-XL (Wang et al., 2026a)	320	700M	1.93	–	298.0	0.80	0.60
JiT-G (Li & He, 2026)	600	2.0B	1.82	–	292.6	0.79	0.62
PixelU-H/16 (Guo et al., 2026)	600	1.17B	1.63	5.04	305.9	0.79	0.64
DiP-XL/16 (Chen et al., 2026b)	600	631M	1.79	4.59	281.9	0.80	0.63
FREPix-XL (Lin et al., 2026)	320	674M	1.91	4.59	295.6	0.79	0.62
DeCo-XL/16 (Ma et al., 2026)	800	682M	1.62	4.41	301	0.80	0.62
PixelDiT-XL/16 (Yu et al., 2026)	80	797M	2.36	5.11	282.3	0.80	0.57
PixelDiT-XL/16 (Yu et al., 2026)	320	797M	1.61	4.68	292.7	0.78	0.64
PixelDiT-XL/16**+WiD**	80	805M	2.09	4.73	278.7	0.79	0.60
PixelDiT-XL/16**+WiD**	200	805M	1.72	4.51	295.4	0.78	0.64
PixelDiT-XL/16**+WiD**	260	805M	1.61	4.67	294.2	0.77	0.65

5 Experiments

5.1 Class-to-Image Generation

Setup. We choose PixelDiT-XL/16(Yu et al., 2026) as the backbone to evaluate our Wiener-informer Decoding on class-conditional ImageNet 256 × 256 (Deng et al., 2009) generation. Following its original training recipe, we use AdamW optimizer (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) with the learning rate of 1e-4 for the first 160 epochs, and then step down to 1e-5. The global batch size of 256 and REPA using DINOv2-B (Oquab et al., 2024) with weight 0.5 remain the same.

As is stated in Section 3.1, we use 51,200 images (about 4% of the training set) to compute the spatial patch statistics to initialize WiD. The computation costs us only **1 minute** with one GPU and 12 CPU DataLoader workers, which we consider quite practical in large-scale training scenarios.

We generate 50K samples from EMA weights using 100-step FlowDPMSolver (Xie et al., 2024) and classifier-free guidance (CFG) (Ho & Salimans, 2021) with interval (Kynkäänniemi et al., 2024), and report FID (Heusel et al., 2017), sFID (Nash et al., 2021), IS (Salimans et al., 2016), precision, and recall (Kynkäänniemi et al., 2019) using the ADM evaluation suite (Dhariwal & Nichol, 2021).

Main results. WiD consistently improves the generation quality of PixelDiT-XL/16 across different training durations. As shown in Table 1, PixelDiT-XL with WiD reaches an FID of 2.09 at 80 epochs, outperforming the reported 2.36 of native PixelDiT at the same duration. Furthermore, WiD reaches FID 1.61 at 260 epochs, matching PixelDiT-XL at 320 epochs with 18.75% fewer training epochs.

The improvement requires only a modest increase in model size (around 1%) from 797M to 805M parameters. Together with the B-size comparisons in Section 4.3, these results demonstrate that our WiD provides consistent improvement and acceleration across different model scales.

Table 2: Text-to-image generation at 256×256 after 80K training steps. We evaluate both methods on 553 GenEval prompts and 1,065 DPG-Bench prompts, with four images per prompt. Higher is better for all scores. WiD improves the performance of DeCo-XXL/16 on both benchmarks.

Method	Steps	GenEval							DPG-Bench
		Sin.Obj.	Two.Obj.	Counting	Colors	Pos.	Color Attr.	Overall ↑	Average ↑
DeCo-XXL/16	80K	0.85	0.32	0.27	0.74	0.12	0.24	0.43	73.1
DeCo-XXL/16+**WiD**	80K	0.92	0.45	0.33	0.81	0.15	0.27	0.49	78.8

5.2 Text-to-Image Generation

Setup. We choose DeCo-XXL/16 (Ma et al., 2026) as the backbone to evaluate our Wiener-informer Decoding on Text-to-Image generation task, since it uses open-source BLIP3-o (Chen et al., 2025a) training data. For an efficient validation, we train DeCo-XXL/16 and DeCo-XXL/16+**WiD** for 80K steps on BLIP3-o pretraining mixture (around 36M image–text pairs), with 256×256 resolution.

Following the original setting of DeCo, we use AdamW with learning rate 2e-4, global batch size 1536, and REPA weight 0.5. We also compute statistics over only 51,200 images (around 1‰of the training set), including 26,400, 19,200 and 5,600 images from Long-Caption, Short-Caption and JourneyDB datasets. We use Adams-2nd solver with 25 steps for sampling, and report the performance on GenEval (Ghosh et al., 2023) and DPG-Bench (Hu et al., 2024) benchmarks.

Main results. As shown in Table 2, WiD consistently improves the performance on both GenEval and DPG-Bench benchmarks at the same training duration. WiD requires only a modest increase of 14M parameters, approximately 1.3% of the 1.1B-parameter baseline.

These results demonstrate that WiD is not only effective across different types of decoder architectures, but also applicable to large-scale text-to-image generation tasks to improve training efficiency.

5.3 Ablations

In this section, we conduct ablation experiments of WiD with PixelDiT-B/16 on ImageNet 256×256, to further support claims and analysis in Section 3&4.

A statistical starting point counts. We first remove the Wiener prediction, which serves as a direct pathway of coarse statistical velocity estimation, retaining the analysis and synthesis coordinate interactions. This leads to the degradation of FID from 41.71 to 49.54, supporting the benefit of the direct statistical prediction pathway. Secondly, we try random initialization of WiD: initializing U_{enc} and U_{dec} with a random orthogonal matrix, and all spectral values λ to 1. Results in Table 3 also support the benefit of statistical knowledge at initialization.

Coordinates for decoder computation matter. Removing the coefficient interfaces between the Wiener and pixel branches increases the FID from 41.71 to 50.67. Moreover, the ablation that removes Wiener gain still outperforms native PixelDiT-B/16. These results support our claim that the statistical coordinates are also beneficial for decoder learning.

Table 3: Ablation experiments on WiD components and initialization on ImageNet 256×256. Using PixelDiT-B as the backbone, we train each model for 40 epochs with REPA and report FID, sFID, and IS evaluated on 50K samples from EMA weights without CFG. Removing the Wiener prediction, coefficient interface or statistics initialization leads to the reduction of generation quality, demonstrating the effectiveness of components and initialization strategy in our WiD.

Method	FID ↓	sFID ↓	IS ↑
PixelDiT-B/16	60.01	14.10	24.89
PixelDiT-B/16+**WiD**	41.71	9.18	35.14
w/o Wiener prediction	49.54	10.88	29.93
w/o Statistics init.	49.02	9.85	30.35
w/o Coefficient interface	50.67	9.14	28.97

6 Related Work

6.1 Statistical Denoising

Statistical priors for image restoration. Wiener estimation has a long history in image denoising and restoration (Wiener, 1949). Methods such as BLS-GSM (Portilla et al., 2003) and NL-Bayes (Lebrun et al., 2013) build on Wiener estimation using statistics for image denoising. Similar to statistical operations in our WiD, BM3D (Dabov et al., 2007) organizes computation by constructing representations, applying empirical Wiener filtering to coefficients, and reconstructing the image through inverse transforms. Furthermore, Deep Wiener Deconvolution (Dong et al., 2020) has proposed to integrate Wiener operations into a learnable feature-processing pipeline for image deblurring. Together, these studies show how statistical priors can be constructed and utilized.

Statistical structure in diffusion. In studies of diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021), statistical estimates also provide a way to understand learned denoisers. Linear fits reveal Gaussian structure associated with generalization (Li et al., 2024), while Gaussian score approximations capture moderate and high noise-level behavior and help accelerate sampling (Wang & Vastola, 2024). Data statistics can also explain locality patterns otherwise attributed to architecture (Lukoianov et al., 2025). These findings connect classical estimation to modern generative models, and inspire us to explore using statistical priors in generative learning.

From estimation to generative learning. Motivated by these insights, our research takes a step further to study how this statistical knowledge can aid pixel-space velocity learning: WiD uses the statistics to initialize the prediction path as a structured starting point, and integrates the computational operations and coordinates into the decoder.

6.2 Pixel-space Flow Models

Prediction parameterization. Recent approaches ease pixel-space learning by changing what the network predicts. JiT (Li & He, 2026) predicts clean images, while AsymFlow (Chen et al., 2026a) combines full-rank data prediction with low-rank noise prediction to reconstruct the velocity. Recently, k-Diff (Jin & Wang, 2026) learns a scalar that controls the data–noise mixture predicted by the network, while retaining velocity-space supervision. A simplified linear analysis motivates this parameterization by relating the preferred mixture to data dimensionality. Separately, WiD aims to address the similar learning difficulty within velocity prediction.

Pixel decoding. Another direction is to separate contextual modeling from dense pixel prediction. PixNerd (Wang et al., 2026a) uses patch-wise neural fields, PixelDiT (Yu et al., 2026) uses pixel transformers, and DeCo (Ma et al., 2026) uses context-conditioned MLP decoders to alleviate the learning difficulty. WiD builds on this separation: it retains the native backbone and decoder blocks in PixelDiT and DeCo, adding statistical interfaces around their existing computation.

Connection to preconditioning. Preconditioning addresses differences in scale and geometry that can hinder diffusion and flow training. EDM (Karras et al., 2022) normalizes network inputs and residual targets using noise-dependent scaling and a skip connection. Preconditioned Flow Matching (Ahamed et al., 2026) instead uses reversible data transformations to improve the conditioning of intermediate flow distributions. WiD is related in the use of statistical structure to ease learning, but focuses on decoder design: a trainable Wiener prediction path interacts with the native decoder.

7 Conclusion

We present Wiener-informed Decoding (WiD) to improve the velocity learning of pixel-space flow models. Specifically, we have shown that coarse second-order statistics can ease pixel-space training, and the gains from frozen Wiener bases extend beyond their explicit linear predictions. Motivated by these insights, we developed WiD, which combines a trainable statistical prediction path with coefficient interfaces that connect it to the native pixel decoder. We demonstrate that WiD enhances training efficiency and stability while consistently improving generation quality across model scales and decoder architectures, benefiting both class-conditional and text-to-image generation with modest parameter overhead. These findings suggest that statistical denoisers can contribute beyond the estimate: the computational organization is also helpful for learning pixel-space generation.

AI USE STATEMENT

In this work, we used generative AI tools to assist providing feedback on research methodology and experiments, implementing code for methods and figure drawing, language translation and polishing. We have not used generative AI tools for synthesizing datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, provide critical ingredients for proving mathematical claims, writing proofs, proposing or refining hypotheses, cleaning and reformating dataset, supporting qualitative and thematic data analysis, or interpreting results. and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. We have verified the accuracy and reliability of code implementation. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Firstly, our experiments are conducted on publicly available datasets, and we strictly follow the preprocessing procedures of studies our work is based on. Moreover, we provide a comprehensive description of our method (Section 3&4), implementation details and training settings (Section 5 and Appendix A.2). Finally, we provide detailed evaluation settings and results (Appendix A.3). We will release our training code, model checkpoints, training logs, and statistical priors to support reproduction and follow-up research after acceptance.

REFERENCES

Shadab Ahamed, Eshed Gal, Md Shahriar Rahim Siddiqui, Simon Ghyselincks, Moshe Eliasof, and Eldad Haber. Preconditioned flow matching, 2026. URL https://arxiv.org/abs/2603.02337.

N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784.

Michael Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025. URL http://jmlr.org/papers/v26/23-1605.html.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Hansheng Chen, Jan Ackermann, Minseo Kim, Gordon Wetzstein, and Leonidas Guibas. Asymmetric flow models. *arXiv preprint arXiv:2605.12964*, 2026a. URL https://arxiv.org/abs/2605.12964.

Jiuhai Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, Le Xue, Caiming Xiong, and Ran Xu. Blip3-o: A family of fully open unified multimodal models-architecture, training and dataset, 2025a. URL https://arxiv.org/abs/2505.09568.

Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow, 2025b. URL https://arxiv.org/abs/2504.07963.

Zhennan Chen, Junwei Zhu, Xu Chen, Jiangning Zhang, Xiaobin Hu, Hanzhen Zhao, Chengjie Wang, Jian Yang, and Ying Tai. Dip: Taming diffusion models in pixel space. In *CVPR*, pp. 36136–36146, June 2026b.

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16 (8):2080–2095, 2007. doi: 10.1109/TIP.2007.901238.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *NeurIPS*, 2021.

Jiangxin Dong, Stefan Roth, and Bernt Schiele. Deep wiener deconvolution: Wiener meets deep learning for image deblurring. In *NeurIPS*, 2020.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. In *NeurIPS*, 2023.

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

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.379. URL https://aclanthology.org/2020.findings-emnlp.379/.

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 *NeurIPS*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment, 2024.

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

Qing Jin and Chaoyang Wang. Revisiting diffusion model predictions through dimensionality, 2026. URL https://arxiv.org/abs/2601.21419.

Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear estimation*. Prentice Hall information and system sciences series. Prentice Hall, Upper Saddle River, NJ, 2000. ISBN 9780130224644.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *NeurIPS*, 2023.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *NeurIPS*, 2019.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. In *NeurIPS*, 2024.

M. Lebrun, A. Buades, and J. M. Morel. A nonlocal bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013. doi: 10.1137/120874989.

Jiachen Lei, Keli Liu, Julius Berner, Y HoiM, Hongkai Zheng, Jiahong Wu, and Xiangxiang Chu. There is no VAE: End-to-end pixel-space generative modeling via self-supervised pre-training. In *ICLR*, 2026.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *CVPR*, pp. 36115–36125, June 2026.

Xiang Li, Yixiang Dai, and Qing Qu. Understanding generalizability of diffusion models requires rethinking the hidden gaussian structure. In *NeurIPS*, 2024.

Mingfeng Lin, Jiakun Chen, Liang Han, and Liqiang Nie. Frepix: Frequency-heterogeneous flow matching for pixel-space image generation, 2026. URL https://arxiv.org/abs/2605.06421.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Artem Lukoianov, Chenyang Yuan, Justin Solomon, and Vincent Sitzmann. Locality in image diffusion models emerges from data statistics. In *NeurIPS*, 2025.

Zehong Ma, Longhui Wei, Shuai Wang, Shiliang Zhang, and Qi Tian. Deco: Frequency-decoupled pixel diffusion for end-to-end image generation. In *CVPR*, pp. 43600–43610, June 2026.

Charlie Nash, Jacob Menick, Sander Dieleman, and Peter Battaglia. Generating images with sparse representations. In *ICML*, 2021.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, pp. 4195–4205, October 2023.

J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003. doi: 10.1109/TIP.2003.818640.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016.

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

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 *ICLR*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Binxu Wang and John Vastola. The unreasonable effectiveness of gaussian score approximation for diffusion models and its applications. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=I0uknSHM2j.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. In *ICLR*, 2026a.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *CVPR*, 2026b.

Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications*. The MIT Press, 08 1949. ISBN 9780262257190. doi: 10.7551/mitpress/2946.001.0001. URL https://doi.org/10.7551/mitpress/2946.001.0001.

Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformers, 2024. URL https://arxiv.org/abs/2410.10629.

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.

Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. In *CVPR*, 2026.

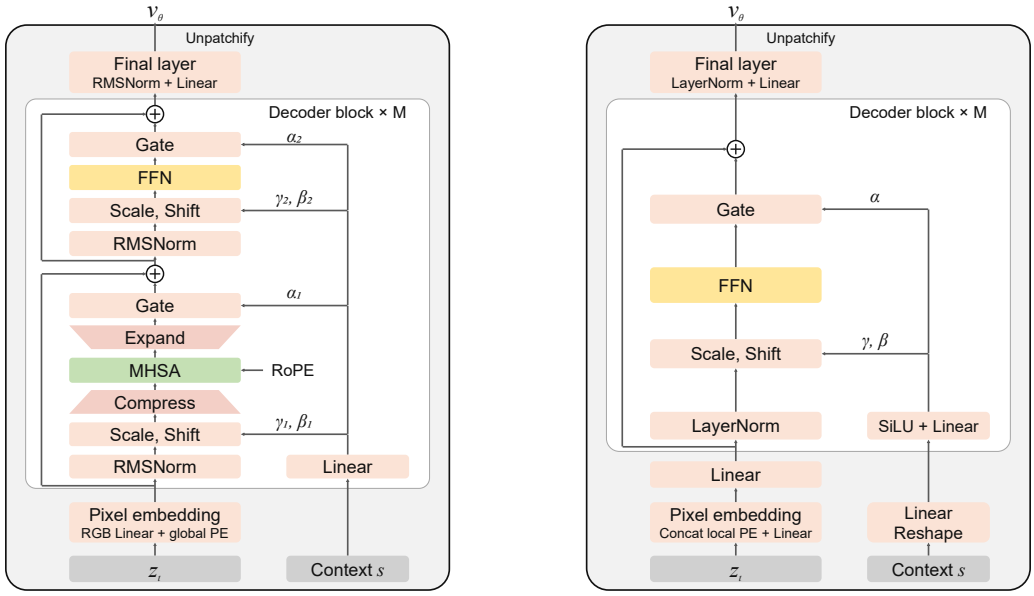
Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019.

Lei Zhang, Weisheng Dong, David Zhang, and Guangming Shi. Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognition*, 43(4):1531–1549, 2010. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2009.09.023. URL https://www.sciencedirect.com/science/article/pii/S0031320309003677.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *ICLR*, 2026.

A Additional Experimental Details

A.1 Decoder Structures in PixelDiT and DeCo



(a) Decoder structure of PixelDiT.

(b) Decoder structure of DeCo.

Figure 5: Illustration of decoder structures in PixelDiT and DeCo.

In this work, we use two representative pixel decoders from PixelDiT (Yu et al., 2026) and DeCo (Ma et al., 2026) to validate our Wiener-informed Decoding approach. As shown in Figure 5, both decoders process the noisy input z_t, use the contextual features s from the backbone to compute scale, shift and gate for pixel-wise modulation, and finally project normalized features to the velocity v_θ. Specifically, PixelDiT decoder involves Multi-Head Self-Attention (MHSA) and Feed-Forward Network (FFN) (Vaswani et al., 2017) blocks, while DeCo decoder mainly uses MLP layers.

Input initialization for WiD on DeCo. The initialization in Section 4.2 preserves each decoder's native positional encoding. PixelDiT adds global positional embeddings after projecting RGB values, whereas DeCo jointly projects RGB values and patch-local cosine positional features. For pixel q in patch i, DeCo's original embedding can be written as $E_0 z_{t,i,q} + E_{\text{pos}}\phi(q) + b$, where E_0 contains the RGB columns of the joint projection and positional information $\phi(q)$ is shared across patches. We initialize its RGB projection and the coefficient expander in WiD as

$$[E_0,\ E_{\text{pos}}] \longmapsto [\tfrac{1}{2}E_0,\ E_{\text{pos}}], \qquad W_F = \tfrac{1}{2}\mathcal{L}(E_0)U_0, \tag{15}$$

leaving the bias unchanged. With the full-rank orthogonal basis U_0, the injected coefficients recover the other half of the RGB contribution, as in equation 14. The summed features therefore match the original embedding before entering native DeCo decoder blocks at the initial state.

Output initialization and normalization for decoders with WiD. Both architectures zero-initialize the coefficient-correction and native pixel-output projections, including their biases. Output normalization follows the native decoder: in DeCo, both heads receive features from its existing per-pixel, non-affine LayerNorm (Ba et al., 2016); in PixelDiT, the coefficient head uses per-pixel RMSNorm (Zhang & Sennrich, 2019), matching the native output head's normalization.

A.2 Additional Implementation Details

Table 4: Detailed settings of the five model configurations used in this work. Parameter counts refer to the native denoising networks, excluding WiD, REPA projectors and external encoders.

	DiT-B	**PixelDiT-B**	**PixelDiT-XL**	**DeCo-B**	**DeCo-XXL**
Architecture					
DiT depth N	12	12	26	12	16
Decoder depth M	–	2	4	3	3
Hidden size D	768	768	1152	768	1536
Attention heads	12	12	16	12	24
Pixel hidden size	–	16	16	32	32
Patch size p	16	16	16	16	16
Native params. (M)	131.4	184.7	797.4	136.0	1133.8
Representation alignment					
Alignment layer	–	8	8	8	6
REPA loss weight	–	0.5	0.5	0.5	0.5
Alignment encoder	–	DINOv2-B	DINOv2-B	DINOv2-B	DINOv2-B
Optimization					
Training data	ImageNet	ImageNet	ImageNet	ImageNet	BLIP3-o
Global batch size	256	256	256	256	1536
Time sampling	Logit-normal	Logit-normal	Logit-normal	Logit-normal	Logit-normal
Training time shift	1.0	1.0	1.0	1.0	2.0
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW
Adam (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.95)$
EMA decay	0.9999	0.9999	0.9999	0.9999	0.9999
Condition dropout	0.1	0.1	0.1	0.2	0.1
Gradient clipping	1.0	1.0	$1.0 \rightarrow 0.5$	1.0	1.0
Learning rate	1e-4	1e-4	1e-4$\rightarrow$1e-5	1e-4	2e-4
Weight decay	0	0	0	0	0
FreqFM loss w.	–	–	–	1.0	1.0
Inference					
Sampler	FlowDPMSolver	FlowDPMSolver	FlowDPMSolver	Euler	Adams-2nd
Sampling steps	100	100	100	50	25
Sampling time shift	1.0	1.0	1.0	1.0	3.0

Training settings. We provide further details of model configurations in Table 4. When validating frozen Wiener bases and WiD, we strictly follow the original training and inference settings of the native denoising networks. Particularly, our DiT-B/16 baseline uses PixelDiT-B's 12-block contextual backbone, including RMSNorm, QK normalization (Henry et al., 2020), 2D RoPE (Su et al., 2023), and SwiGLU (Shazeer, 2020), rather than the original DiT block design. We replace the pixel decoder with a single conditioned patch-wise velocity head and use PixelDiT's pixel-space flow-matching recipe. This construction allows us to compare direct velocity prediction with pixel decoding while keeping the backbone architecture unchanged.

Reproducibility. Our PixelDiT-XL/16+WiD and DeCo-XXL/16+WiD are trained on 8 NVIDIA H200 GPUs. For reproducibility, we fix the random seeds for native-model initialization and for sampling the 51,200 images used to estimate the patch PCA statistics. We also save each rank's random-number-generator (RNG) state in training checkpoints and restore it when resuming interrupted runs. We will release our training code, model checkpoints, training logs, and statistical priors to support reproduction and follow-up research.

A.3 Additional Evaluation Details

For class-to-image generation on ImageNet-1K 256×256, we strictly follow the native sampling and evaluation protocol of each original backbone. For comparisons in Figure 1& 4, we generate 10K images for each method without classifier-free guidance (CFG) and report FID-10K results. For results of PixelDiT-XL/16+WiD reported in Table 1, we use CFG with guidance interval, and we provide more results and detailed settings in Table 5 here. At 160 and 200 training epochs, our method also outperforms the native PixelDiT-XL/16 which officially reports FIDs of 1.97 and 1.94.

Table 5: We provide more results of PixelDiT-XL/16+WiD with different CFG weights and intervals.

Method	Epochs	CFG	Interval	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
PixelDiT-XL/16	80	3.25	[0.10, 1.00]	2.36	5.11	282.3	0.80	0.57
PixelDiT-XL/16	320	2.75	[0.10, 0.90]	1.61	4.68	292.7	0.78	0.64
PixelDiT-XL/16+**WiD**	80	2.90	[0.10, 1.00]	2.09	4.73	278.7	0.79	0.60
PixelDiT-XL/16+**WiD**	160	2.60	[0.10, 1.00]	1.87	4.50	280.1	0.79	0.62
PixelDiT-XL/16+**WiD**	200	2.75	[0.10, 0.90]	1.72	4.51	295.4	0.78	0.64
PixelDiT-XL/16+**WiD**	260	2.75	[0.10, 0.90]	1.65	4.58	296.4	0.78	0.65
PixelDiT-XL/16+**WiD**	260	3.00	[0.12, 0.90]	1.64	4.69	295.1	0.77	0.65
PixelDiT-XL/16+**WiD**	260	3.25	[0.12, 0.70]	1.61	4.67	294.2	0.77	0.65

Reproducibility. We sample 50K images using FlowDPMSolver-100 with the same seed 1600 as the official setting of PixelDiT-XL/16, and use the same ADM evaluation protocal to compute the scores. Therefore, the results are reproducible with our open-source code and model checkpoints.

Computational overhead. On PixelDiT-XL/16, WiD increases the denoiser parameter count from 797.38M to 804.86M and the analytical forward cost from 311.19 to approximately 315.32 GFLOPs, corresponding to increases of 0.94% and 1.33%, respectively. We follow PixelDiT's linear-layer and attention counting convention and additionally account for WiD's explicit matrix products.

We benchmark denoiser-only inference on a single NVIDIA H200 using EMA weights and BF16, with six warm-up iterations followed by 24 synchronized wall-clock measurements. At batch size four, WiD reduces forward throughput from 238.8 to 227.6 samples/s for PixelDiT-XL and from 260.2 to 245.1 samples/s for our reproduced DeCo-XXL/16 baseline.

With a 25-step Adams-2nd sampler and CFG 4.0, batch-four text-to-image generation throughput is 7.50 images/s for the DeCo-XXL/16 baseline and 7.04 images/s for DeCo-XXL/16+WiD.

B Additional Diagnostic Details

We expand the five-time averages in Figure 2. The analyses below are mainly about frozen Wiener bases and native pixel decoding, before introducing adaptive WiD.

B.1 Comparison of Prediction Errors

Evaluation protocol. We use the official PixelDiT-XL epoch-320 EMA checkpoint as the prediction reference and evaluate B-size models at 10, 20, 30 and 40 training epochs. At each $t \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$, we fit on 4,096 ImageNet validation images and score on 3,072 disjoint images, with two fixed noise draws per image. All comparisons share the same inputs, class labels, and reference predictions, using EMA weights without CFG; the B-size models are trained without REPA. These diagnostics mainly measure the agreement with a well-trained model.

Table 6: Time-resolved remaining-error reductions, in units of 10^{-3}. Positive values indicate closer agreement with the reference after removing the fitted affine responses. Each block specifies its comparison baseline; all use patch-affine fits except DiT+DCT.

Epoch	$t = 0.1$	$t = 0.25$	$t = 0.5$	$t = 0.75$	$t = 0.9$
DiT+PCA relative to plain DiT					
10	6.519	9.084	15.104	22.932	19.754
20	5.585	8.244	13.326	19.678	15.777
30	5.874	8.368	12.840	18.729	15.142
40	5.718	8.211	12.547	18.246	14.880
DiT+DCT relative to plain DiT					
10	8.955	11.067	16.279	24.144	25.817
20	7.653	9.980	14.353	20.131	17.621
30	7.043	9.389	13.563	19.176	17.251
40	6.945	9.264	13.267	18.694	17.030
Native PixelDiT relative to plain DiT					
10	4.719	7.812	13.172	18.363	18.547
20	4.929	7.612	12.648	19.150	22.380
30	5.455	7.957	12.423	18.706	20.593
40	5.653	8.266	12.862	19.951	26.588
PixelDiT+PCA relative to native PixelDiT					
10	1.661	1.369	2.232	6.207	11.998
20	0.734	0.815	1.111	2.490	5.884
30	0.506	0.615	0.895	2.191	7.909
40	0.186	0.196	0.250	0.719	2.745

Fitted responses and remaining predictions. At each fixed t, we fit an affine map to each model's (including the reference) complete velocity output. The PCA family is a shared 768×768 patch-affine map with bias; the DCT family consists of independent 3×3 RGB-affine maps with biases at each of 256×256 frequencies.

Each affine family contains its corresponding frozen base, so subtracting the fitted response removes that kind of base's explicit additive contribution. Note that the remaining prediction can still contain linear dependencies outside the chosen family, such as cross-patch or cross-frequency interactions.

Error accounting. Let $e_{\mathrm{lin}} = \ell_f - \ell_{\mathrm{ref}}$ and $e_{\mathrm{rem}} = (f - \ell_f) - (f_{\mathrm{ref}} - \ell_{\mathrm{ref}})$. The errors satisfy

$$\begin{aligned} E_{\mathrm{tot}}(f) &= E_{\mathrm{lin}}(f) + E_{\mathrm{rem}}(f) + E_{\mathrm{cross}}(f), \\ E_{\mathrm{cross}}(f) &= \frac{2}{d}\mathbb{E}\langle e_{\mathrm{lin}}, e_{\mathrm{rem}}\rangle. \end{aligned} \tag{16}$$

For a baseline b and a comparison model f, we report $\Delta E = E(b) - E(f)$. Therefore, positive gains over total, linear, or remaining errors indicate smaller errors relative to the reference.

Gains beyond the additive prediction. Table 6 reports remaining-error reductions at each time and training epoch. Both frozen bases and native pixel decoding improve the remaining prediction over plain DiT. These gains do not require uniformly better fitted linear responses. For example, at 40 epochs and $t = 0.9$, DiT+DCT and native PixelDiT have higher linear errors but lower remaining and total errors. All models share the DiT backbone design and global batch size of 256.

Additional gains with pixel decoding. The last block compares PixelDiT with and without a frozen PCA base. The base remains useful with a native decoder to 40 epochs: all five remaining-error gains are positive. They shrink overall with training, consistent with the decreasing FID gains in Figure 1c.

B.2 Pathway Responses to Input Perturbations

Pathway decomposition. We probe native PixelDiT-B using 256 fixed held-out images, one noise per image, EMA weights, and no CFG. With t, y fixed, automatic differentiation gives the two output-velocity JVPs in equation 10. The pixel pathway holds the contextual features at their current values; and the contextual pathway holds the decoder's direct noisy input fixed. The reference here is the analytic response $w = A_t(\widetilde{M})\delta$ for either Wiener base.

Comparing pathway alignment. Following Figure 2c, we remove each response's component parallel to the isotropic unit perturbation δ before comparison:

$$j^{\perp} = j - \frac{\langle j, \delta \rangle}{\|\delta\|^2}\delta, \qquad w^{\perp} = w - \frac{\langle w, \delta \rangle}{\|\delta\|^2}\delta. \tag{17}$$

We compute cosine similarities per image and report the mean difference between the pixel and contextual pathways. Positive values favor the pixel pathway alignment with the reference.

Table 7: Time-resolved differences in pathway alignment with Wiener responses. Entries are the mean cosine to the reference for the direct pixel pathway minus that for the contextual pathway.

Epoch	$t = 0.1$	$t = 0.25$	$t = 0.5$	$t = 0.75$	$t = 0.9$
PCA reference					
10	$\mathbf{-0.137}$	0.200	0.405	0.316	0.292
20	$\mathbf{-0.101}$	0.261	0.514	0.439	0.403
30	$\mathbf{-0.054}$	0.307	0.548	0.458	0.430
40	0.048	0.400	0.591	0.481	0.431
DCT reference					
10	$\mathbf{-0.351}$	0.027	0.273	0.254	0.272
20	$\mathbf{-0.313}$	0.091	0.376	0.373	0.380
30	$\mathbf{-0.259}$	0.136	0.409	0.394	0.410
40	$\mathbf{-0.150}$	0.228	0.452	0.418	0.412

Dependence on noise level. Table 7 favors the pixel pathway at all four evaluated times from 0.25 to 0.9. At $t = 0.1$, however, the contextual pathway shows generally closer alignment with Wiener bases to DCT at every checkpoint and to PCA at 10–30 epochs. Averaging over all five times, the pixel pathway has higher direction-removed cosine for each reference.

C Limitations and Future Work

In this work, we mainly focus on easing velocity learning for pixel-space image generation tasks. Actually, we think the perspective of statistical denoising may also be helpful for other flow generative models. Therefore, explorations of integrating statistical denoisers into other generative tasks are also attractive directions, such as latent space generation, and language, audio, as well as video generation. In the future, we are planning to improve upon our current approach, and take a step further to explore the potential of integrating Wiener-informed structures directly into diffusion transformers, and try to develop next foundation architecture for flow-matching tasks.

D ADDITIONAL VISUALIZATIONS



Figure 6: 256×256 samples of PixelDiT-XL/16+WiD at 260 epochs. We use classifier-free guidance with $w = 4.0$. Class label = "arctic wolf" (270).



Figure 7: 256×256 samples of PixelDiT-XL/16+WiD at 260 epochs. We use classifier-free guidance with $w = 4.0$. Class label = "baseball" (429).



Figure 8: 256×256 samples of PixelDiT-XL/16+WiD at 260 epochs. We use classifier-free guidance with $w = 4.0$. Class label = "cheeseburger" (933).



Figure 9: 256×256 samples of PixelDiT-XL/16+WiD at 260 epochs. We use classifier-free guidance with $w = 4.0$. Class label = "cliff drop-off" (972).