PIXELDDT: DEPTH-ROUTED DECODING FOR PIXEL DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pixel-space diffusion generates images directly in pixel space without a learned image autoencoder. Recent decoupled architectures pair a global transformer with a lightweight pixel decoder, aiming to separate global semantic modeling from local detail rendering. However, pixel-decoder architectures remain insufficiently compared under a shared implementation. We first compare representative decoder families in a shared codebase and identify a compact multiscale U-Net as a competitive design. This architectural split, however, does not determine how the two modules share information. The pixel-decoder receives backbone information only through the final transformer state at its bottleneck. Probes show that this state retains coarse and fine image content, while class separation declines across later backbone blocks. Earlier states, on the other hand, already expose visual content with stronger class separation, motivating direct access to these features at different decoder scales. We therefore introduce PixelDDT, which routes distinct backbone depths to successive U-Net stages while retaining the final-state bottleneck. These connections allow local multiscale reconstruction to draw on earlier features without passing through the terminal backbone state. With matched backbone and decoder widths, depth routing improves ImageNet-256 FID from 2.74 to 2.31 at 400K updates. Gated window attention further exchanges features across neighboring patches, bringing FID to 2.27 at the same training budget. With longer training, PixelDDT achieves FID 1.63 after approximately 380 epochs with classifier-free guidance.

1 INTRODUCTION

Pixel-space diffusion generates images without a learned image autoencoder, but must process a larger signal than latent diffusion (Ho et al., 2020; Dhariwal & Nichol, 2021; Rombach et al., 2022; Peebles & Xie, 2023; Ma et al., 2024). Decoupled architectures address this cost by pairing a global patch transformer with a smaller pixel decoder: coordinate-conditioned MLPs in PixNerd, patch U-Nets in DiP, residual pixel MLPs in DeCo, and pixel-level transformers in PixelDiT (Wang et al., 2025a; Chen et al., 2026; Ma et al., 2026; Yu et al., 2026). Beyond efficiency, this separation suggests a division of labor between global conditioning and local rendering. DeCo makes this objective explicit by separating semantic conditioning from high-frequency detail generation (Ma et al., 2026).

An architectural split, however, does not determine how the two modules share information. In the DiP-style patch U-Net, the backbone supplies its terminal state at the decoder bottleneck. Our probes show that the terminal backbone state retains information about both coarse image structure and fine details (Figure 3(a)), while layerwise probes show declining class separation across later backbone blocks due to this burden. On the other hand, earlier states already expose visual content with stronger class separation. Giving decoder stages direct access to these states could reduce their dependence on the terminal path and let local multiscale computation perform the remaining reconstruction.

We introduce *PixelDDT*, a decoupled pixel diffusion model with depth-routed conditioning and cross-patch refinement. We compare decoder families in a shared codebase and choose a compact multiscale U-Net. Alongside its final-state bottleneck, depth routing connects deeper backbone states directly to coarse decoder stages and shallower states to finer stages. The fine stages thus

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.

receive earlier visual features without requiring them to pass through the terminal backbone state and every preceding decoder stage. Each stage combines its assigned backbone representation with its recursive input and local skip. All source states occupy the same token grid; the receiving decoder supplies the spatial scales. These paths allow the backbone and decoder to adapt their division of labor during joint training.

With matched backbone and decoder widths, depth routing lowers FID from 2.74 to 2.31. The interface adds 0.087% of denoiser parameters and 1.98% of forward FLOPs over bottleneck-only conditioning. Probes show how the learned organization changes: class separation increases at middle backbone depths, while the decoder's clean-image readouts and frequency sensitivities shift. The terminal backbone still retains detail content. These observations describe the representations learned with direct depth access; generation quality is evaluated separately through FID.

Depth routing also leaves a spatial interface to consider: patchwise decoders receive global context but cannot directly exchange neighboring decoder activations. PixelDDT adds gated regular and shifted-window attention over the reassembled feature grid, obtaining FID 2.27 at 400K updates. With longer training, it reaches ImageNet-256 FID 1.63 after approximately 380 epochs. Together, the depth and spatial connections organize how global conditioning and local pixel processing cooperate in a jointly trained model.

2 Related Work

Pixel-space generation and local decoders. Diffusion models and their training schedules establish strong pixel-space generation (Ho et al., 2020; Dhariwal & Nichol, 2021; Karras et al., 2022; Hoogeboom et al., 2023); hierarchical transformers provide another route to efficient spatial modeling (Crowson et al., 2024). Decoupled designs include neural-field decoding in PixNerd (Wang et al., 2025a), the convolutional patch U-Net in DiP (Chen et al., 2026), frequency-aware training with a lightweight decoder in DeCo (Ma et al., 2026), and patch/pixel transformer hierarchies in PixelDiT (Yu et al., 2026). PixelFlow studies pixel-space flow generation (Chen et al., 2025). DiP's detailer comparison already includes standard and coordinate MLPs, intra-patch attention, and a U-Net. We compare representative decoder families before studying the conditioning interface of a multiscale U-Net.

Representations and their decoding interfaces. U-Net uses local encoder skips to support dense reconstruction (Ronneberger et al., 2015). Dense Prediction Transformers combine intermediate states of a constant-resolution transformer through a multiscale convolutional decoder (Ranftl et al., 2021); ControlNet injects spatial conditions into a diffusion network (Zhang et al., 2023). These designs establish access to intermediate features as a general principle. We test it in a jointly trained patchwise velocity predictor against a U-Net conditioned only at its bottleneck. REPA shapes intermediate generative features using a pretrained representation (Yu et al., 2025; Oquab et al., 2024); PixelREPA and register-token work further study the organization of pixel-diffusion representations (Shin et al., 2026; Starodubcev et al., 2026). In our 29-block models, the shallowest source is also the REPA-aligned block. The experiments do not separate access to several depths from direct access to aligned features.

Prediction targets and spatial refinement. JiT and PixelU study clean-data prediction (Li & He, 2026; Guo et al., 2026), while DDT examines semantic/velocity decoupling (Wang et al., 2025b). Our analysis uses velocity prediction. Shifted windows connect local partitions (Liu et al., 2021); PixSGR combines a supervised low-channel bottleneck with coarse-guided sparse global refinement (You et al., 2026). Our cross-patch module instead refines intermediate patch-U-Net features, and gives adjacent patches a direct interaction before the final upsampling stage. Guidance changes quality/diversity tradeoffs (Ho & Salimans, 2022); we keep its sampling recipe fixed within the guided controlled comparisons.

3 Comparing Pixel Decoder Architectures

We study class-conditional ImageNet-256 generation (Russakovsky et al., 2015) with the time convention $x_t = tx_0 + (1-t)\epsilon$ and velocity target $v^* = x_0 - \epsilon$ (Lipman et al., 2023; Liu et al.,

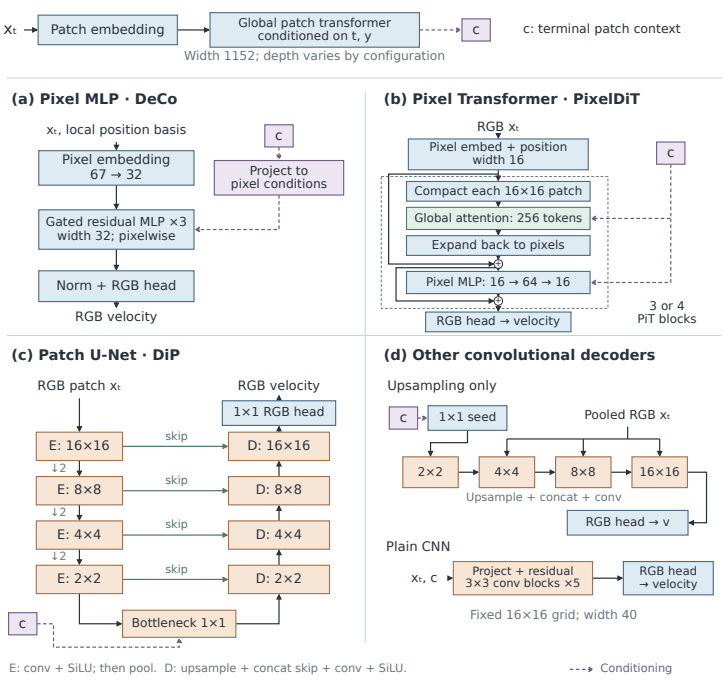

Figure 1: **Pixel-decoder families in the shared implementation.** Terminal patch context c conditions each reconstruction path. DeCo mixes channels pointwise; PixelDiT attends over compacted patch tokens; DiP uses a four-stage U-Net with local skips. The remaining variants use an upsampling path without encoder skips or a constant-resolution CNN. Sizes are patch-local; outputs are reassembled into RGB velocity. Table 1 gives the complete-denoiser parameter counts.

2022). A global transformer computes $h_1, \ldots, h_L$ from the noisy image, time, and class. The pixel decoder combines the noisy pixels with a backbone condition to predict RGB velocity. Clean-image reconstructions in the interface analysis are diagnostic readouts, distinct from this training target.

Published pixel-space systems differ in their codebases, training recipes, model sizes, and samplers, making their final scores insufficient to isolate the pixel decoder. We therefore implement the decoder families in the DeCo codebase and compare them after 400K updates on the same ImageNet-256 task, using a fixed guided sampling recipe and documented unguided archive settings. Backbone and decoder capacity are allocated within a similar full-denoiser compute range; Table 1 reports the complete-denoiser parameter counts. Clipping and accumulation differ across configurations, as documented in Appendix A.1.

Figure 1 compares pixel MLPs following DeCo, pixel transformers following PixelDiT, patch U-Nets following DiP, and other convolutional decoders. The MLP mixes channels at individual pixels. PixelDiT combines pixel-wise transformations with global attention over compacted patch features. The U-Net downsamples each noisy patch, retains local encoder skips, and reconstructs it through four decoder stages. The remaining convolutional alternatives use an upsampling path without encoder skips or a constant-resolution CNN. Comparing several configurations within each family tests how these structural choices perform under the shared implementation and compute scale.

The DiP reproduction reaches FID 18.68 without guidance amplification and 2.45 with CFG, making the U-Net a competitive starting point within the measured decoder families. Its four reconstruction scales also expose natural receiving sites for backbone features, which pointwise MLPs and constant-resolution CNNs do not provide.

In DiP's patch U-Net, the local encoder aggregates noisy-patch information at multiple resolutions, while skips deliver its intermediate features directly to corresponding decoder stages. With the backbone condition fixed, isolated-path probes show a predominantly pixel-local response through the finest skip and broader within-patch responses through coarser paths (Appendix A.2). These distinct, overlapping responses suggest that local reconstruction draws on different spatial scales.

Table 1: Pixel-decoder families at 400K updates on ImageNet-256. Backbone width is 1152; parameter counts cover the complete denoiser. Guided FID uses 50K images. Further settings appear in Appendix A.1.

		No CFG			CFG
Variant and configuration	Params (M)	FID↓	sFID↓	IS↑	FID↓
Pixel MLP (DeCo family)					
DeCo reproduction (28 DiT + 3 MLP)	682.28	21.74	7.08	64.3	2.76
MLP (29 DiT + 3 MLP)	706.18	20.21	7.36	67.9	2.55
Pixel Transformer (PixelDiT family)					
PixelDiT (20 DiT + 4 PiT; width 16)	653.99	20.14	7.59	68.5	2.66
PixelDiT (23 DiT + 3 PiT; width 16)	682.60	19.50	7.56	68.0	2.49
PixelDiT (24 DiT + 3 PiT; width 16)	706.50	19.12	7.23	**70.2**	2.50
U-Net (DiP family): 4 encoder + 4 decoder stages					
DiP reproduction (28 DiT + U-Net; base 48)	678.24	**18.68**	5.83	69.7	**2.45**
Narrower U-Net (29 DiT + U-Net; base 36)	699.85	19.81	5.82	67.3	2.74
Other pixel-space decoders					
Decoder only (29 DiT + U-Net Decoder; 4 stages, base 48)	699.37	20.07	5.76	66.1	2.56
Inverted hourglass (28 DiT + InvHourglass; 4 stages, base 48)	675.47	20.29	**5.57**	67.2	2.73
Plain CNN (29 DiT + 5 CNN Blocks)	697.20	49.18	14.26	38.2	15.04

Together, the encoder and skips combine hierarchical local processing with direct reuse of local features. We therefore choose this family for the conditioning-interface analysis.

4 Architectural Separation and Learned Representations

The canonical decoupled architecture separates global processing from pixel prediction, but a terminal-only interface still asks one backbone state to supply all decoder scales. We examine this issue in the multiscale patch U-Net selected above. It combines the raw noisy patch processed by its local encoder with the global transformer condition supplied at the bottleneck. Let L_s denote the local encoder skip at stage s, R_s the upsampled recursive input, and D_s the output after fusion convolution and SiLU. The first recursive input already combines the deepest local feature and the final transformer condition. Thus R_0 is a mixed decoder state, rather than the last transformer block itself.

Earlier features provide an alternative to terminal-only conditioning. In the DiP reproduction, null-label class-probe accuracy falls from 57.6% at block 8 to 36.9% at block 15 and 15.9% at its final block 28. Earlier states already carry clean-image information: the conditional $D64$ gain is .190 at block 8 and .443 at block 15, reaching .564 at the terminal state. Here $D64$ denotes the clean Haar detail band at resolution 64, and conditional gain measures prediction beyond the noisy patch and position (Section 5.2). The coexistence of accessible visual features and stronger class separation at earlier depths motivates providing them directly to the decoder.

Different paths expose different content. We freeze the generative model and fit affine heads to its features at $t = .5$ with null class labels. The stage-native readout predicts the clean image average-pooled to that stage's reassembled resolution: $32, 64, 128$, or 256. Its R^2 therefore compares paths against the same target within a stage. In the bottleneck-only U-Net, the coarse local skip has higher clean-image readability than the recursive input (.969 versus .857). At resolution 128, the recursive input becomes more image-readable than the skip (.922 versus .875). The relative clean-image accessibility of the two paths therefore changes across decoder scale.

Division of labor depends on feature access. The path readouts in Appendix B.1 show that routing changes recursive and fused representations while local skips remain similarly readable. We therefore organize conditioning around direct access to several backbone depths at the decoder scales that combine global and local features. The following probes characterize the learned representations, and Section 7 evaluates generation.

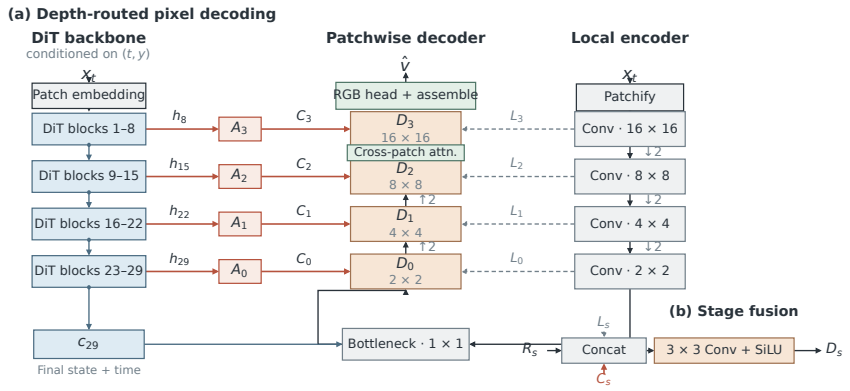

Figure 2: **PixelDDT architecture.** The bottleneck-only baseline retains the local encoder, skips, recursive decoder, and final-state bottleneck. PixelDDT adds the red depth-to-scale projections and cross-patch attention after stage 2. Block numbers are one-based; stage sizes are patch-local.

5 Depth-Routed Conditioning

5.1 Connecting backbone depth to decoder scale

We retain the final-state bottleneck and add four projected connections to the U-Net decoder (Figure 2). Deeper states condition coarse stages, while shallower states reach fine stages for local refinement. At stage s, a learned affine projection A_s maps a selected backbone state to 64 channels; $\mathcal{B}_s$ broadcasts each patch token over its local grid. The stage fuses this condition with its upsampled recursive input and local skip:

$$C_s = \mathcal{B}_s\big(A_s h_{\pi(s)}\big)\,, \qquad D_s = \mathrm{SiLU}(W_s * [R_s, L_s, C_s] + b_s)\,. \tag{1}$$

The recursive input is upsampled before concatenation; W_s is a patch-local 3×3 convolution. Figure 2 specifies the source depths and stage resolutions. A final 1×1 convolution predicts RGB velocity.

Each source is projected to the same conditioning width before fusion. This gives every stage a direct backbone connection without changing the local encoder, skip widths, or output widths. Appendix A accounts for the projections, additional fusion channels, and complete parameter and FLOP budgets.

All tapped backbone states share the same 16×16 token grid. The spatial pyramid belongs to the U-Net receivers; routing is a prescribed depth-to-scale connection, without input-dependent selection. The separate bottleneck uses the final condition $c_{29} = \mathrm{SiLU}(t_{\mathrm{emb}} + h_{29})$ together with the deepest local feature. The tapped states already depend on time and class through the backbone; the bottleneck applies this additional time embedding and activation.

5.2 Backbone and decoder representations

We probe the final backbone output and local decoder at matched widths, using frozen 400K EMA weights, $t = .5$, and null labels (Figure 3(a)–(b)). The backbone feature X is the raw 1152-dimensional block-29 token before bottleneck projection. For each clean Haar band, we fit two independent affine ridge heads on the same training patches: f_B receives the noisy RGB patch and its position, while $f_{B,X}$ additionally receives X. Both predict the same clean-band coefficients Y:

$$\mathrm{SSE}_B = \sum_{(i,p)\in\mathcal{V}} \|f_B(B_{ip}) - Y_{ip}\|_2^2, \quad \mathrm{SSE}_{B,X} = \sum_{(i,p)\in\mathcal{V}} \|f_{B,X}([B_{ip}, X_{ip}]) - Y_{ip}\|_2^2, \tag{2}$$

where $\mathcal{V}$ contains all patches of the validation images. We define the conditional gain as

$$G = 1 - \frac{\mathrm{SSE}_{B,X}}{\mathrm{SSE}_B} = \frac{\mathrm{SSE}_B - \mathrm{SSE}_{B,X}}{\mathrm{SSE}_B}. \tag{3}$$

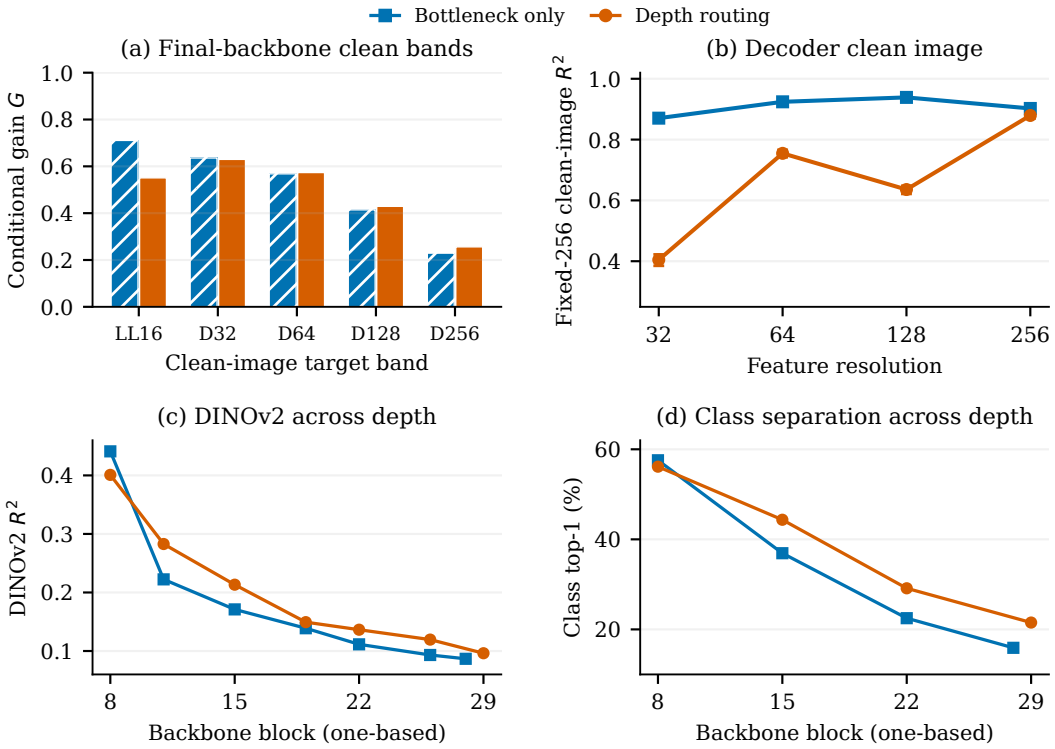

Figure 3: **Representation readouts.** (a) Conditional clean-band gains from raw backbone block 29. (b) Decoder clean-image readouts with 95% test-image intervals. (c) DINOv2 patch-token readouts. (d) Null-label class top-1. All use frozen 400K EMA weights and $t = .5$. Panels (a)–(b) match backbone and decoder widths. Panels (c)–(d) compare the 28-block/base-48 DiP reproduction with 29-block/base-36 depth routing, both seed 0. DINO and class probes use 1,024/256 and 50K/10K train/validation images. Appendix B gives protocols; Appendix B.2 reports the matched conditioning-source ablation.

It is the fraction of the baseline's remaining prediction error removed by adding X. Thus $G = .711$ means a 71.1% reduction relative to the noisy-patch baseline. Each head selects its regularization separately; Appendix B specifies the targets, splits, and fitting procedure.

The terminal representations retain clean-image information across frequencies. Patch-mean gain is .711 for bottleneck-only conditioning and .551 for depth routing, while the finest-detail gain is .229 and .256, respectively. The intermediate detail bands remain close. This profile shows a change in the content accessible at the final backbone state, alongside the additional intermediate sources exposed by routing. Appendix B.2 gives an additional control for the choice of source depth.

Decoder heads predict the same full 256×256 clean image from every stage, with identical head geometry and image/noise splits across models. Routing lowers early-stage R^2 from .870 to .405; final-stage values are .902 and .879. Thus the better generation in Table 2 accompanies lower intermediate clean-image readability. Later decoder stages can combine their inputs without requiring an early affine reconstruction. The probes characterize how representations change when the decoder gains direct access to intermediate backbone states.

Semantic accessibility along the backbone. Figure 3 combines the matched final-state and decoder probes with layerwise DINOv2 and class readouts from the recorded backbone study. Relative to the bottleneck-only DiP reproduction, depth routing retains higher semantic readability through the middle and later blocks: at block 15, DINO-target R^2 is .213 versus .171, and class top-1 is 44.4% versus 36.9%. Both profiles decline after the aligned block 8. Terminal class accuracy is still 15.9% for DiP and about 21.5% for routing, well above the 0.1% chance level. Panels (a)–(b) use

matched widths, whereas panels (c)–(d) retain the architecture difference of the recorded layerwise study.

6 Direct Cross-Patch Refinement

Depth routing determines which backbone states reach each decoding scale. PixelDDT also introduces a spatial connection: adjacent patches share global transformer context, but their local decoder features are otherwise processed independently. We exchange these features before the final reconstruction stage.

Connecting local decoder features. The local U-Net processes each 16×16 image patch separately, so a convolution at a patch edge cannot access the neighboring patch's decoder activations. PixelDiT reduces the channel dimension before global self-attention over compacted patch features, but the interaction remains global over the spatial token set. We instead retain the compact U-Net and localize attention in the spatial dimension, directly connecting neighboring patch features at bounded cost.

An intermediate spatial connection. After fusion at the local 8×8 stage, we reassemble a 128×128 feature grid. Global self-attention on its 16,384 positions would introduce a quadratic spatial cost. Instead, we apply gated regular and half-window-shifted attention (Liu et al., 2021):

$$\widetilde{D} = D + g(c_{29}) \odot \big(\mathcal{A}_{\text{regular}}(D) + \mathcal{A}_{\text{shifted}}(D)\big). \tag{4}$$

Both branches use windows of side 16 and share normalization, attention, and output projections. Each $\mathcal{A}$ includes its zero-initialized output projection. The per-patch scalar gate is $g(c) = \sigma(W_g\,\mathrm{SiLU}(c) + b_g)$, initialized with $W_g = 0$ and $b_g = -2$. The result returns to the patch layout before the final upsampling stage. This cross-patch (x-patch) module adds 22,033 parameters and 3.775G FLOPs; Figure 2 shows its placement. PixelDDT combines depth routing and this spatial connection in one jointly trained denoiser.

Both partitions span neighboring feature patches. For $N = 128^2$ positions and $M = 16^2$ positions per window, their combined attention-matrix cost is $2NM$, a factor of 32 below global attention's N^2. Appendix A illustrates the parallel branches and gives their full budget.

7 Generation Results and Interface Controls

7.1 Training and Evaluation

The 400K, 29-block controls share global batch 256, AdamW at 10^{-4}, BF16 training, EMA decay .9999, class dropout .2, frequency-weighted velocity loss, and REPA at block 8 (Ma et al., 2026; Yu et al., 2025; Oquab et al., 2024); each endpoint is one run. Guided comparisons use the released DeCo recipe: EMA, Euler-100, time shift 1, and CFG scale 3.2 on $(.1, 1]$ (Ma et al., 2026; Ho & Salimans, 2022). We generate 50K indexed images, 50 per class, and evaluate FID with the ADM pipeline (Heusel et al., 2017; Dhariwal & Nichol, 2021). Appendix A gives the loss and budget details; Appendix C specifies numeric display and sampling protocols. Representation readouts use frozen checkpoints and do not run a generative sampler.

7.2 Depth Routing and Spatial Interaction

Direct access to backbone depth. Table 2 evaluates the two connections in PixelDDT. Direct depth access lowers FID from 2.74 to 2.31, while retaining the local encoder, skip connections, decoder widths, and final-state bottleneck. Routing adds 606,208 parameters and 5.247G FLOPs. This comparison evaluates the complete conditioning interface, including its additional receiving sites. A source-control ablation repeats the terminal state at the same sites with identical trainable modules, obtaining FID 2.46 versus 2.31 for routing (Appendix B.2).

Adding spatial interaction. Adding x-patch attention to the 29-block routed model reduces FID from 2.31 to 2.27. The model retains the same backbone, depth-routing map, and base-36 decoder.

The spatial module adds 22,033 parameters and 3.775G FLOPs, giving a complete denoiser with 700.480M parameters and 274.621G FLOPs. Frozen-checkpoint interventions below examine how the trained module uses neighboring-patch features.

7.3 Longer training and published systems

Following the DeCo sampling setting at a nominal 200-NFE budget, PixelDDT reaches FID 2.29 at approximately 80 epochs and FID 1.63 at approximately 380 epochs (Table 3). The same long-trained checkpoint gives FID 1.71 under the fixed Euler recipe used for the controlled ablations. Appendix C gives the exact sampler settings and evaluation accounting.

Quality at the reported training exposure. Figure 4 compares FID against single-forward TFLOPs times epochs. PixelDDT reaches FID 1.63 at exposure 104.4, versus FID 1.69 at 147.0 for 600-epoch DeCo. Each system retains its reported sampler. The exposure axis combines model arithmetic and training duration; it omits backward, optimizer, and auxiliary-encoder costs and does not measure wall-clock training time.

Table 2: **PixelDDT component ablation at 400K with CFG.** All rows share backbone and decoder widths. Depth routing adds backbone connections; x-patch connects neighboring decoder features. Appendix A gives complete model budgets.

Configuration	FID↓
DiP (Bottleneck only)	2.74
+ Depth routing	2.31
+ X-patch (PixelDDT)	**2.27**

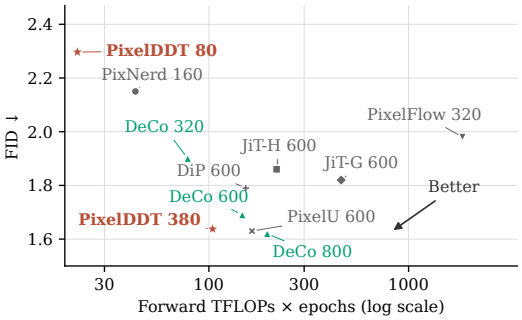

Figure 4: **FID versus forward-compute exposure.** Lower left is preferred; each point uses the reported sampling recipe.

Table 3: ImageNet-256 systems under reported sampling recipes. NFE includes both CFG branches; GFLOPs use the two-operation convention. * Profiled; † converted; – not reported. Details appear in Appendix C.

Method	M params	Epochs	NFE	GFLOPs	FID↓	sFID↓	IS↑
PixNerd-XL/16 (Wang et al., 2025a)	700	160	200	268	2.15	4.55	297.0
PixelFlow-XL/4 (Chen et al., 2025)	677	320	240	5818	1.98	5.83	282.1
DiP-XL/16 (Chen et al., 2026)	631	600	200	255*	1.79	4.59	281.9
JiT-H/16 (Li & He, 2026)	953	600	200	364†	1.86	–	303.4
JiT-G/16 (Li & He, 2026)	2000	600	200	766	1.82	–	292.6
DeCo-XL/16 (Ma et al., 2026)	682	320	200	245*	1.90	4.47	303.0
DeCo-XL/16 (Heun) (Ma et al., 2026)	682	600	200	245*	1.69	4.59	304.0
DeCo-XL/16 (Ma et al., 2026)	682	800	500	245*	**1.62**	**4.41**	301.0
PixelU-H/16 (Guo et al., 2026)	1168	600	200	274†	1.63	5.04	**305.9**
PixelDDT (Ours)	700	80	200	275	2.29	5.07	272.7
PixelDDT (Ours)	700	**380**	200	275	1.63	4.64	303.1

7.4 Functional effects of the decoder connections

Frequency-selective responses. The readouts describe accessible content, while feature interventions test how the trained decoder responds to that content. We split each post-fusion feature map into low- and high-frequency DCT components on the reassembled spatial grid. At one stage at a time, we subtract a component rescaled to 5% or 10% of the full feature norm, then replay the

remaining decoder. Both directions therefore have equal perturbation energy. The model weights, encoder skips, and backbone inputs remain fixed. We measure the resulting change in the predicted clean image on 256 shared validation images at $t = .5$ with null labels. The perturbed velocity prediction gives the clean-image estimate $\widehat{x}_0 = x_t + .5\widehat{v}$. Responses are measured relative to the same checkpoint's unperturbed output, whereas the clean-target errors below use the ground-truth image. This distinction separates sensitivity of the learned computation from the accuracy of its reconstruction. Appendix B.4 gives the mask and paired-bootstrap protocol.

Let Q be the squared output change from the low-frequency intervention divided by that from the high-frequency intervention. At 5% perturbation, Q changes from 1.580 to .190 at the coarsest stage and from 2.158 to .838 at the next stage when depth routing is introduced. The corresponding routing-to-baseline ratios are .120 and .388, with 95% image-bootstrap intervals $[.114, .128]$ and $[.376, .400]$. The direction persists at 10% perturbation. Early routed stages thus respond less strongly to the low-frequency direction relative to the equally sized high-frequency direction. At the final stage, both models have $Q < .06$, with the ordering reversed (.038 versus .053). These functional responses complement the lower early-stage clean-image readability, locating the largest changes before the final reconstruction stage.

Clean-target band errors show where these changes affect reconstruction. Under a 10% low-frequency perturbation, patch-mean SSE at the first three stages increases by 23.2%, 74.9%, and 128.5% for bottleneck-only conditioning, compared with .6%, 3.9%, and 8.2% for routing. At the final stage, the increases are close: 71.2% and 73.9%. The routed decoder is therefore less dependent on the perturbed low-frequency components at early and middle stages, while the final reconstruction still depends on them. Without perturbation, its clean-target SSE is approximately 99% of the baseline across all five Haar bands. The changed sensitivity profile coexists with a small, broadly distributed improvement in reconstruction accuracy rather than a selectively larger gain in the finest band.

Use of neighboring-patch features. We mask only attention edges that cross the original patch partition, preserving the learned normalization, projections, gate, and regular and shifted-window branches. At 400K updates, this raises velocity MSE by .091%, .381%, and 1.138% at $t = .2, .5, .8$, respectively, over 1,000 shared validation images with null labels. The increase is positive at all three noise levels at 100K and 200K as well (Appendix C.1).

A separate connectivity test fixes the backbone conditions and encoder skips, then perturbs one patch's intermediate 8×8 decoder feature by 1% of its norm using a central finite difference. The native x-patch module produces an output response outside the perturbed patch. Neither the patch-wise routed decoder nor the edge-masked operator does so. Holding the backbone inputs fixed separates this new decoder connection from the global context already present in the transformer features. Combined with edge masking, the test establishes both the spatial reach of the implemented operator and its use by the trained checkpoint.

Bypassing the whole module has a different effect from removing its cross-patch edges: at 200K and $t = .5$, velocity MSE rises by 60.86% for the former and .402% for the latter. The module combines neighboring-feature exchange with within-patch transformations, so its total contribution cannot be attributed to cross-patch edges alone. These measurements characterize dependence within frozen checkpoints; Table 2 separately measures generation quality after joint training. Together with the frequency interventions, they distinguish changes in the trained computation from changes in what an affine probe can reconstruct.

8 Conclusion

PixelDDT connects distinct backbone depths to a multiscale pixel decoder and exchanges features across neighboring patches. At matched backbone and decoder widths, these connections improve FID over bottleneck-only conditioning. Probes reveal changes in class separation and clean-image accessibility, with fine detail retained in the terminal backbone. PixelDDT reaches ImageNet-256 FID 1.63 after approximately 380 epochs under a nominal 200-NFE guided evaluation following the DeCo sampling setting. The learned division of labor depends on how global and local features meet, beyond assigning their computation to separate backbone and decoder modules.

AI USE STATEMENT

Generative AI tools were used in a supporting capacity during parts of the research and writing process, including research planning, implementation support, analysis of experimental results, and manuscript preparation. All reported results derive from the experiments described in the paper, and the authors reviewed and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Sections 5 and 7 specify the model interfaces, training recipe, and generation evaluation. Appendix A gives architecture budgets, Appendix B specifies the readout heads and splits, and Appendix C distinguishes the sample protocols. The appendix also records the numerical values and protocols underlying the manuscript figures; no separate supplementary file is required to interpret the reported evidence.

REFERENCES

Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow, 2025. 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, 2026. URL https://arxiv.org/abs/2511.18822.

Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z. Kaplan, and Enrico Shippole. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 9550–9575. PMLR, 2024. URL https://proceedings.mlr.press/v235/crowson24a.html.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

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, 2026. URL https://arxiv.org/abs/2606.27760.

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. URL https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images, 2023. URL https://arxiv.org/abs/2301.11093.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022. URL https://arxiv.org/abs/2206.00364.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise, 2026. URL https://arxiv.org/abs/2511.13720.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

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.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. URL https://arxiv.org/abs/2103.14030.

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, 2024. URL https://arxiv.org/abs/2401.08740.

Zehong Ma, Longhui Wei, Shuai Wang, Shiliang Zhang, and Qi Tian. Deco: Frequency-decoupled pixel diffusion for end-to-end image generation, 2026. URL https://arxiv.org/abs/2511.19365.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12179–12188, October 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Ranftl_Vision_Transformers_for_Dense_Prediction_ICCV_2021_paper.html.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. URL https://arxiv.org/abs/1505.04597.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. URL https://arxiv.org/abs/1409.0575.

Jaeyo Shin, Jiwook Kim, and Hyunjung Shim. Representation alignment for just image transformers is not easier than you think, 2026. URL https://arxiv.org/abs/2603.14366.

Nikita Starodubcev, Ilia Sudakov, Ilya Drobyshevskiy, Artem Babenko, and Dmitry Baranchuk. Registers matter for pixel-space diffusion transformers, 2026. URL https://arxiv.org/abs/2605.16147.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion, 2025a. URL https://arxiv.org/abs/2507.23268.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer, 2025b. URL https://arxiv.org/abs/2504.05741.

Weiyi You, Jinhua Zhang, Xingyu Zhou, Wei Long, Junyu Lou, and Shuhang Gu. Advanced pixel diffusion model with guided sparse global refinement, 2026. URL https://arxiv.org/abs/2609.00798.

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, 2025. URL https://arxiv.org/abs/2410.06940.

Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation, 2026. URL https://arxiv.org/abs/2511.20645.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2302.05543.

A Architecture and Budget Details

The coarse-to-fine routing map is $\pi = (29, 22, 15, 8)$ with one-based block indices. Patch-local grids of side $2, 4, 8, 16$ reassemble to image grids of side $32, 64, 128, 256$, with output widths $288, 144, 72, 36$. Backbone and decoder configurations appear in Table 1; Appendix A.1 summarizes their operators and training differences. All use backbone width 1152 and effective batch 256.

The matched conditioning-interface models use velocity matching plus the inherited frequency-weighted residual loss (coefficient 1) and REPA alignment (coefficient .5) to DINOv2-B/14 at backbone block 8 (Ma et al., 2026; Yu et al., 2025; Oquab et al., 2024). Frequency weighting uses the inverse JPEG quantization weights at quality 85, with exponent 1.

The four conditioning projections and wider fusion convolutions add 0.606M parameters and 5.247G FLOPs (0.087% and 1.98%) over bottleneck-only conditioning. Denoiser parameter counts exclude the training-only REPA projector, pretrained DINOv2 encoder, and EMA copy.

Forward FLOPs are profiled on one 256×256 image, including attention matrix multiplications and counting a multiply-add as two operations. They measure model arithmetic, not wall-clock latency.

Table 4: The evaluated backbone-to-decoder interfaces. Block numbers are one-based; stages run from coarse to fine. The last two rows have identical trainable modules and receiving locations.

Interface	Bottleneck source	Extra receiving stages	Sources at stages 0–3
Bottleneck only	29	None	–
Final broadcast	29	0, 1, 2, 3	29, 29, 29, 29
Depth routing	29	0, 1, 2, 3	29, 22, 15, 8

All controlled interface and spatial comparisons use the 29-block, base-36 configuration selected in a preliminary unguided search. The 28-block, base-48 DiP reproduction has a different capacity allocation. Depth routing uses 700.458M parameters and 270.845G FLOPs; adding x-patch attention gives 700.480M and 274.621G.

Figure 5 shows how regular and shifted windows on the reassembled grid connect neighboring patches before the final decoder stage.

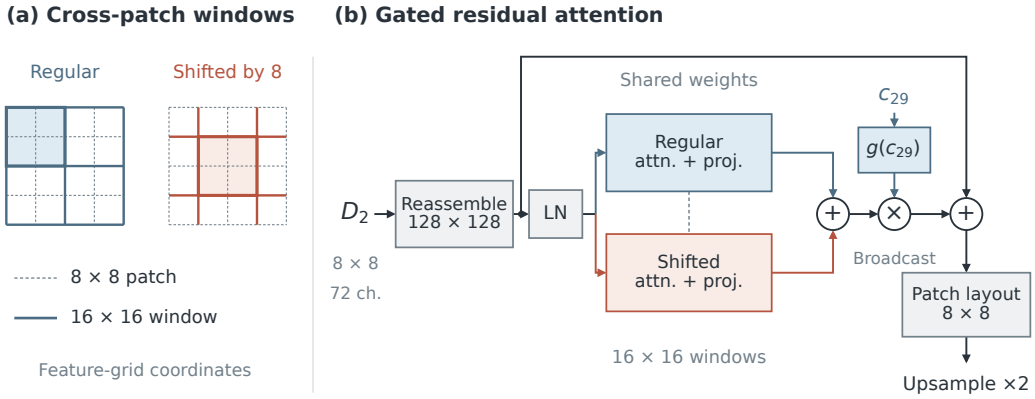

Figure 5: **Cross-patch feature interaction.** Left: a 16×16 attention window spans multiple 8×8 feature patches, with a second partition shifted by eight positions. Right: the two attention branches share parameters and act in parallel. Shifted attention returns its result to the original coordinates within the operator. Their sum is gated by the final backbone condition and added to the original feature grid; the next decoder stage upsamples the restored patch features.

A.1 Pixel-decoder family configurations

The ten configurations in Table 1 use width-1152 backbones and 16×16 image patches.

Decoder operators. The pixel MLP uses three width-32 residual blocks with SiLU, pixel-wise AdaLN, and NeRF RGB embeddings. DiP U-Nets have four encoder and decoder stages, widths $(8b, 4b, 2b, b)$ for base width b, and terminal-state bottleneck conditioning; the narrower variant uses $b = 36$. Decoder-only and inverted-hourglass variants upsample a terminal-state seed with pooled noisy-RGB inputs and no convolutional encoder. The plain CNN uses five width-40 residual blocks with 3×3 convolutions at full patch resolution.

Pixel-transformer operators. PiT blocks combine width-16 pixel features with global attention over compacted patch representations and pixel-wise MLPs. The terminal backbone state conditions each block through shifts, scales, and gates; spatial information is supplied by positional embeddings and rotary attention encodings.

Training and evaluation. All configurations use the DeCo objectives and common optimizer settings in Section 7, with seed 0 and effective batch 256. PixelDiT uses two-step gradient accumulation and clipping at 1; the remaining models use one-step accumulation, with clipping only for the inverted-hourglass variant. Guided sampling follows Section 7; unguided sampling uses guidance 1. PixelDiT retains historical 50K-image archives, while the inverted-hourglass model uses indexed samples in both regimes. IS uses ten splits after a fixed random permutation. Inverted-hourglass FLOPs come from a configuration profile.

A.2 Native decoder locality and multiscale paths

We probe the seed-0, 400K DiP checkpoint on 1,000 class-balanced ImageNet validation images with fixed noise and null labels at $t = .5$. Small central finite-difference perturbations of a pixel isolate each local encoder path while holding backbone conditioning and other paths fixed. Table 5 decomposes the output responses into Haar bands.

The finest skip concentrates its response in $D256$, while coarser paths span overlapping bands. These responses support multiscale local processing, without establishing a strict frequency partition or a preferred backbone-to-scale assignment.

Table 5: Isolated RGB-input paths in the frozen DiP checkpoint at $t = .5$. Each row retains the RGB-induced change in one encoder output while fixing the other outputs and the backbone condition. Energy is the squared finite-difference response, summed over output pixels and channels and averaged over images. Haar columns are mean per-image fractions (%) of that row's response energy. Different paths are not an additive attribution.

Path	Energy	$LL16$	$D32$	$D64$	$D128$	$D256$
Encoder bottleneck	0.000171	10.9	12.5	16.2	20.3	40.1
2^2 skip	0.000397	10.8	26.7	27.0	14.8	20.8
4^2 skip	0.011188	40.6	1.3	30.2	19.2	8.7
8^2 skip	0.037767	19.2	1.7	21.2	35.8	22.1
16^2 skip	3.642929	0.2	1.0	3.7	16.2	78.8

The summed path response matches the complete local RGB response with relative squared error below 4.4×10^{-6} over the measured images and noise levels; individual energies are not additive because responses can reinforce or cancel.

B Readout and Intervention Details

Shared affine ridge heads reconstruct clean RGB images from frozen decoder features. Progressive readouts use targets pooled to each stage's resolution; fixed-target readouts predict 256×256 images at every stage. Head geometry and decoder widths are matched across conditioning interfaces.

Readouts use 1,024/256/256 fit/tune/test images with independent noise. Fit and tune images are disjoint ImageNet training subsets; testing uses validation images. Features are standardized from the fitting split, and regularization is selected on the tuning split. Heads are fitted on sampled spatial cells and evaluated over all test pixels using 400K EMA checkpoints.

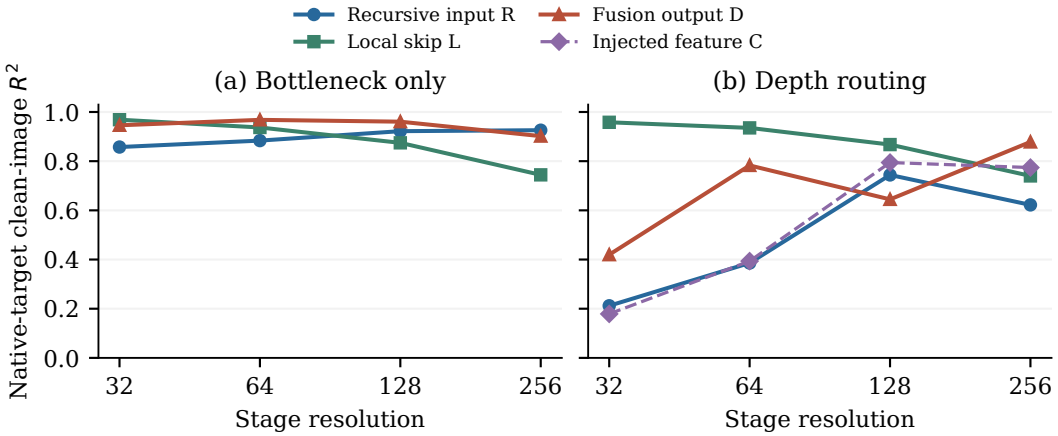

Figure 6: **Path readouts under bottleneck-only conditioning and depth routing.** At each stage, an affine head predicts the clean image at the corresponding reassembled resolution (32^2, 64^2, 128^2, or 256^2). Thus each within-stage comparison uses the same target, while targets differ across stages. Both models use matched widths, 400K EMA, $t = .5$, and null labels. C is the projected backbone feature in Equation 1; the other paths are defined in Section 4.

Backbone readouts. Linear class probes use globally pooled features and class-balanced 50K/10K training/validation splits, with null labels at $t = .5$. All probes use fixed noise realizations.

Conditional clean-band gain. Band readouts use non-overlapping 16×16 patches from 1,024 training and 256 validation images with independent noise. Shared affine ridge heads predict clean-image Haar coefficients from noisy pixels and position, with or without the backbone feature (Equation 2). Decoder features, when used, are averaged within each original patch.

Four orthonormal Haar levels give $LL16$, proportional to the patch mean, and progressively finer detail bands $D32$, $D64$, $D128$, and $D256$. Inputs use training-set standardization; targets remain unstandardized. Each head selects regularization independently for each band. The same validation set is used for selection and reporting; unlike the image readouts, these diagnostics have no separate test split.

DINO readouts predict clean-image DINOv2-B/14 patch tokens. Raw band and DINO R^2 use an X-only head and normalize its SSE by the target's squared deviation from its training-set mean. This differs from G, whose denominator is the fitted noisy-input baseline error.

The terminal-backbone comparison in Figure 3(a) uses raw block-29 outputs before bottleneck projection or added time embedding, with shared images, noise, and ridge protocol. Bottleneck only and depth routing give DINO-target $R^2 = .0871$ and $.0962$, respectively. This comparison changes both receiving sites and sources; the source-depth control below fixes the receiving layers.

Plain-U-Net path probes. Path probes read the recursive input R, local skip L, injected condition C, and fused output D using matched widths and frozen EMA features at $t = .5$ with null labels. The initial recursive state already combines local and final-backbone features; bottleneck-only conditioning has no separate C path.

Heads predict stage-resolution clean images using the same fit/tune/test split as above. Path evaluation samples one cell per original patch, whereas fixed-target evaluation covers every pixel. All interface comparisons share images, noise, and fitting procedures. The raw-input reference is an affine RGB-to-RGB head on x_t, serving as an input anchor rather than a parameter-matched model.

B.1 Bottleneck-conditioned paths and routed representations

In the bottleneck-only model, the local skip is more image-readable than the recursive input at the two coarse resolutions; the ordering reverses at the two fine resolutions (Figure 6). Native-target scores compare paths within each stage, whereas the fixed-256 protocol uses a common target across stages.

B.2 Effect of conditioning-source depth

Final broadcast replaces the four routing sources with the terminal block, $\pi = (29, 29, 29, 29)$, while matching projections, fusion layers, parameters, and FLOPs. Without x-patch, it reaches FID 2.46 versus 2.31 for depth routing and 2.74 for bottleneck-only conditioning (Table 7), testing source selection separately from receiving-site placement.

Under these matched configurations, depth routing improves intermediate class separation, while terminal class accuracy and detail accessibility remain similar.

B.3 Definitions of diagnostic measurements

Clean-image readout. For both the progressive and fixed-target protocols, clean-image readout is scored as

$$R^2 = 1 - \frac{\sum_i \|\widehat{x}_i - x_i\|_2^2}{\sum_i \|x_i - \bar{x}\|_2^2}, \tag{5}$$

with per-channel means over the evaluated pixels or sampled cells specified by each protocol. Scores measure accessibility to that head and target.

Decoder feature spectra. For matched post-fusion grids, an orthonormal 2D DCT of the reassembled $H \times W$ features gives the energy fraction $\sum_{u+v>.25(H+W-2)} E(u,v) / \sum_{u,v} E(u,v)$. Squared coefficients are summed over channels and 10,767 class-balanced validation images, including DC in the denominator and without patch pooling. Stagewise fractions are $.337, .353, .061, .076$ for bottleneck only and $.632, .518, .198, .102$ for routing. These representation-dependent statistics do not measure reconstruction accuracy or denoising work.

B.4 Frequency-selective decoder interventions

We use the two 400K EMA checkpoints, FP32 inference, $t = .5$, null labels, and 256 shared class-balanced validation images with paired noise. Cached decoder replay matches the full-forward velocity elementwise; interventions hold weights, skips, and injected inputs fixed.

For a post-fusion feature map D_s, reassemble the complete spatial grid and use the DCT mask above to form orthogonal low- and high-frequency components D_s^{low} and D_s^{high}. For $b \in \{\text{low}, \text{high}\}$, intervene at one stage using

$$D_s' = D_s - \alpha \frac{\|D_s\|_F}{\|D_s^b\|_F} D_s^b, \qquad \alpha \in \{.05, .10\}. \tag{6}$$

Per-image norms cover all channels and positions, giving equal-energy low/high perturbations. With $\widehat{x}_0 = x_t + .5\widehat{v}$, define $E_s^b = \sum_i \|\widehat{x}_{0,i}^b - \widehat{x}_{0,i}\|_2^2$ and $Q_s = E_s^{\text{low}} / E_s^{\text{high}}$. Model ratios use 2,000 paired image-bootstrap resamples; intervals describe image variability for one checkpoint per model.

Section 7.4 reports the perturbed patch-mean errors. Unperturbed routing-to-baseline SSE ratios are $.990, .992, .993, .992, .990$ across $LL16$ through $D256$, indicating small improvements across all bands. These interventions measure dependence on feature components at one noise level; they neither isolate source depth from added receiving sites nor explain the FID gain by themselves.

C Generation and Evaluation Protocols

Guided short-run comparisons use 50K indexed samples, 50 per ImageNet class, with deterministic noise assignments shared across sampling workers. FID uses all samples. Displayed FIDs are truncated to the shown precision; published results retain their source values.

Table 6: **Frequency-selective feature perturbations.** For each perturbation magnitude, columns report Q for Bottleneck only (B), depth routing (R), and R/B with a 95% paired image-bootstrap interval. All four stages are shown.

	5% feature-norm perturbation			10% feature-norm perturbation		
Stage	B	R	R/B [95% CI]	B	R	R/B [95% CI]
0 (32)	1.580	.190	.120 [.114, .128]	1.701	.193	.114 [.107, .121]
1 (64)	2.158	.838	.388 [.376, .400]	2.199	.838	.381 [.370, .393]
2 (128)	.852	.532	.624 [.602, .647]	.849	.539	.635 [.612, .658]
3 (256)	.038	.053	1.400 [1.380, 1.420]	.038	.053	1.400 [1.380, 1.420]

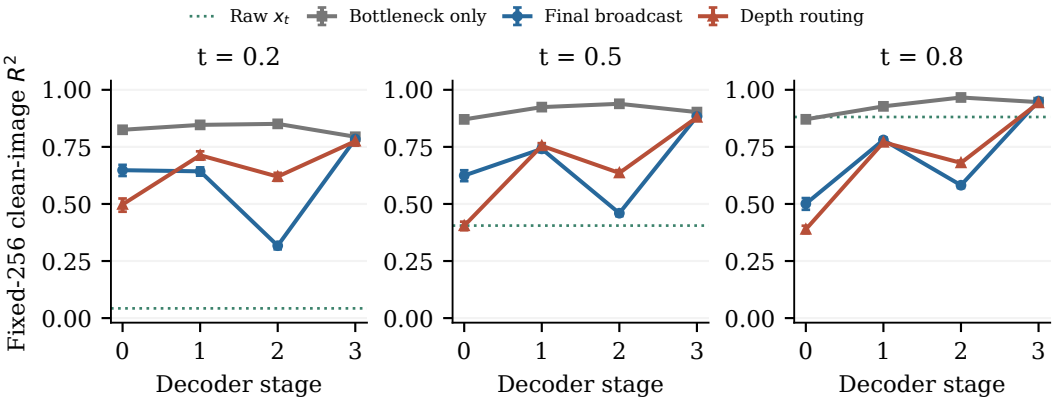

Figure 7: **Fixed-target readouts at matched decoder widths.** Bottleneck only, final broadcast, and depth routing use identical output widths and head dimensions, shared image and noise splits, and frozen 400K EMA checkpoints. Every stage predicts the full 256×256 image. Error bars are 95% test-image bootstrap intervals; the dotted line is a fitted raw-RGB input anchor.

Our guided architecture, interface, and spatial-component comparisons use Euler-100, CFG 3.2 on $(.1, 1]$, and time shift 1. Published system rows preserve the cited papers' reported settings.

System FLOP sources. Table 3 uses profiled forward costs for the released DiP and DeCo architectures and published parameter counts. JiT-H/16 and PixelU-H/16 FLOPs are converted to the two-operation multiply-add convention. Costs describe a single forward pass independently of NFE and guidance; normalization and elementwise operations are not exhaustively counted.

Backbone depth and decoder width. With a 28-block backbone, depth routing achieves FID 2.32 using a narrower base-36 U-Net, compared with 2.45 for the base-48 bottleneck-only decoder (Table 7). This comparison changes both width and conditioning. With 29 blocks and decoder width fixed at base 36, routing improves FID from 2.74 to 2.31.

Table 7: **Depth routing across backbone depths and decoder widths.** ImageNet-256 FID at 400K updates under the fixed guided recipe on 50K images. The 28-block comparison also changes decoder width; the 29-block comparisons keep it fixed.

DiT blocks	U-Net base	Conditioning	FID↓
28	48	Bottleneck only	2.45
28	36	Depth routing	2.32
29	36	Bottleneck only	2.74
29	36	Final broadcast	2.46
29	36	Depth routing	2.31
29	36	Depth routing + x-patch	**2.27**

Long-training comparison. The approximately 380-epoch checkpoint reaches FID 1.63 on 50K images with Heun-50, time shift .9, and CFG 3.3 on $(.1, .95]$. The 80-epoch checkpoint uses the same preselected recipe and reaches FID 2.29 on indexed, class-balanced 50K samples. The longer-run archive predates this indexing protocol. Heun-50 has nominal NFE 200 including both CFG branches, with 198 executed branch evaluations because the final corrector is omitted. The 380-epoch checkpoint gives FID 1.71 under the fixed Euler-100 recipe used for controlled comparisons.

C.1 Use of cross-patch information in frozen checkpoints

We evaluate seed-0, 29-block checkpoints at 100K, 200K, and 400K updates on 1,000 shared class-balanced validation images, with paired noise, null labels, and $t \in \{.2, .5, .8\}$. Masking only attention edges across original patch boundaries preserves within-patch transformations and all learned weights. We report the change in velocity MSE without retraining.

Table 8: Relative increase in velocity MSE after masking only cross-patch attention edges in the frozen 29-block model. Each entry uses 1,000 images and the same fixed checkpoint and inputs as its unmasked reference.

Updates	$t = .2$	$t = .5$	$t = .8$
100K	0.148%	0.410%	0.949%
200K	0.114%	0.402%	1.070%
400K	0.091%	0.381%	1.138%

Cross-patch masking increases error at every evaluated checkpoint and noise level. At 200K and $t = .5$, bypassing the whole module raises MSE by 60.86%, versus 0.402% for edge masking: the module contributes both neighboring-feature exchange and within-patch processing. These interventions measure dependence within frozen checkpoints, not retraining outcomes.

A separate finite-difference check perturbs one patch's intermediate feature by 1% of its norm while fixing backbone conditions and encoder skips. Only the native x-patch model responds outside that patch; the patchwise decoder and edge-masked operator do not. Decoder replay is verified against the complete forward pass.