RESIDUAL-STREAM BURDEN SHAPES REPRESENTATION LEARNING IN DIFFUSION TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

In diffusion-based generation, a neural network can be trained to predict the clean data, the noise, or the velocity from a noisy input. These prediction targets are interconvertible and describe the same generative process, yet plain Diffusion Transformers operating on large pixel patches succeed with clean prediction and fail with noise or velocity prediction. We argue that this asymmetry arises because noisy targets require the residual stream to preserve noise-dependent input variation through depth for the final readout, forcing subsequent layers to compute on noisy representations. A spectrally concentrated clean target imposes a lighter demand, leaving greater freedom to organize hidden representations for subsequent computation. We call this preservation requirement *residual-stream burden* and show how it shapes representation learning in Diffusion Transformers. Controlled experiments indicate that the exploitable structure is spectral concentration in patch space and that the bandwidth of the persistent residual state is a key resource for noisy prediction. We further show that this account is consistent with recent decoupled pixel-space architectures, whose diverse designs all reduce the residual-stream burden on the main pathway. To examine this understanding from a complementary direction, we expand and reorganize the residual-stream bandwidth directly, introducing Spatially Indexed Hyper-Connections (SiHC) that reach FID 1.71 on ImageNet 256^2. Together, these results identify residual-stream burden as a mechanism through which prediction targets and architecture jointly shape representation learning in Diffusion Transformers.

1 INTRODUCTION

Denoising generative models learn a data distribution by training a neural network to separate clean data from its corrupted version. Given a noisy input, one can ask the network to predict the noise ($\boldsymbol{\epsilon}$-prediction) (Ho et al., 2020), the velocity ($\boldsymbol{v}$-prediction, a linear combination of clean data and noise) (Salimans and Ho, 2022; Lipman et al., 2023; Liu et al., 2023), or the clean data itself ($\boldsymbol{x}$-prediction) (Li and He, 2026). These parameterizations describe the same generative process and can be converted into one another at inference time, so one might expect them to present comparable learning problems. However, when plain Diffusion Transformers (DiTs) (Peebles and Xie, 2023) operate on large image patches, the choice becomes decisive: with 16×16 patches, $\boldsymbol{x}$-prediction succeeds while the other two fail catastrophically (Li and He, 2026).

A recent explanation attributes the advantage of $\boldsymbol{x}$-prediction to the *low-dimensional manifold* of natural images, arguing that this structure allows clean prediction to use less *model capacity* while noisy targets force the network to reproduce high-dimensional noise (Li and He, 2026). But this account leaves two key concepts imprecise: what kind of low-dimensionality does the network exploit, what exactly is the capacity consumed by noisy prediction, and how do the two interact during training? We answer these questions and show how the resulting understanding informs architecture design for diffusion models.

Our contributions. We propose that the advantage of $\boldsymbol{x}$-prediction arises because noisy targets require a plain DiT's hidden states to retain noise-dependent input variation through the residual stream for the final readout, so that subsequent layers must compute on noisy representations. As patch size grows, more corruption-dependent pixel variation is packed behind each token, increasing

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.

the preservation pressure on its residual state. By contrast, a clean target concentrated in fewer directions reduces the demand on *information-carrying capacity* for the final readout.

More generally, the output target determines what information must be preserved through depth, and this preservation requirement shapes the representations learned along the residual stream. We call this requirement *residual-stream burden*. Accordingly, the relevant low-dimensionality should appear in the patch space through which a DiT sees the image, and the relevant capacity should be the bandwidth of the persistent state that carries information across depth. Specifically, we find that

- *The exploitable low-dimensionality is spectral concentration in patch space.* We show that the exploitable structure is the spectral concentration of clean-patch variance: an invertible whitening of the patch space that preserves manifold structure but removes this concentration destroys the success of $\boldsymbol{x}$-prediction (Section 4.1). We further show that this concentration shapes representation learning differently under $\boldsymbol{x}$- and $\boldsymbol{v}$-prediction (Section 4.2).
- *The relevant capacity is residual-stream bandwidth.* Using Hyper-Connections (HC) (Zhu et al., 2025a) to expand the persistent residual state $4\times$ while keeping computation nearly unchanged, we rescue $\boldsymbol{v}$-prediction from complete failure, indicating that residual-stream bandwidth is a key resource for accommodating the burden of noisy prediction (Section 5.1).

This understanding also accounts for the success of several recent pixel-space architectures that predict velocity through decoupled designs, such as DeCo, PixelDiT, and DiP (Ma et al., 2026; Yu et al., 2026; Chen et al., 2026b), whose separate pixel paths absorb fine-scale noise prediction and relieve the main backbone's burden. We verify that their backbone streams develop the same concentrated structure and representation-learning behavior as $\boldsymbol{x}$-prediction in a plain DiT (Section 4.3).

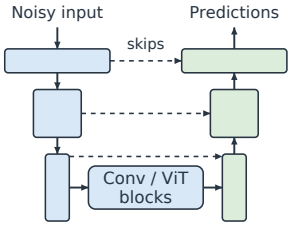
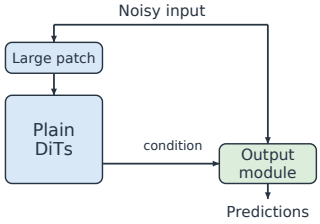
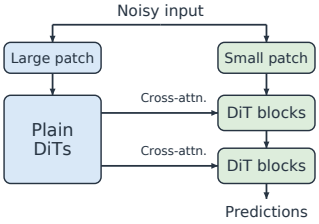
To further examine this understanding, we explore a complementary direction: instead of relieving burden through a separate decoder or switching to $\boldsymbol{x}$-prediction, we reorganize the residual-stream bandwidth itself. We introduce *Spatially Indexed Hyper-Connections* (SiHC), which expand the persistent bandwidth and distribute the prediction burden across local subpatch states. Under direct $\boldsymbol{v}$-prediction without a specialized decoder or cross-attention, SiHC-XL with REPA (Yu et al., 2025) reaches FID 1.71 on ImageNet 256^2, competitive with recent pixel-space methods.

2 Related Work

Decoupled designs in diffusion models. Decoupled designs separate semantic or low-frequency modeling from the fine-scale reconstruction required by the diffusion target, an idea made explicit by DDT through a semantic encoder and a specialized velocity decoder (Wang et al., 2026c). The design has since been adopted in pretrained representation spaces such as RAE (Zheng et al., 2026) and in recent pixel-space models (Table 1). Prediction reparameterization can be viewed as a fixed, analytic form of decoupling, in which $\boldsymbol{x}$-prediction supplies the noisy input through an analytic skip (Li and He, 2026; Chen et al., 2026a). Our analysis shows that these seemingly different designs share the effect of reducing the noise-dependent burden on the main DiT, thereby easing the friction between velocity modeling and representation learning. SiHC achieves this separation without a separate pixel decoder or cross-attention path, expanding and organizing residual-stream bandwidth so that local states carry the burden outside the shared workspace.

Hyper-Connections and virtual width. Hyper-Connections (HC) replace the single residual stream with multiple persistent streams and learned cross-depth connections (Zhu et al., 2025a), allowing representational width to grow independently of computational width, a degree of freedom termed *virtual width* (Seed et al., 2025). Follow-up work has made expanded residual streams stable and scalable, and HC variants now appear in large language models (Zhu et al., 2025b; Xie et al., 2026; Zhang et al., 2026; Z.ai, 2026; Tencent Hy Team, 2026). Their role in generative modeling remains much less understood, and the original HC work reports only a modest improvement on latent DiT-XL/2 (FID $2.36 \to 2.18$) (Zhu et al., 2025a). More recently, Diffusion-Adaptive Routing studies how cross-layer aggregation affects optimization in diffusion Transformers and improves training through timestep-aware access to earlier layer outputs, routing information along depth rather than widening the persistent state (Xu et al., 2026). Our work provides a demand-side view of residual-stream bandwidth from the perspective of residual-stream burden and shows that accommodating this burden with virtual width yields particularly large gains in high-dimensional flow matching.

Table 1: **A taxonomy of pixel-space diffusion architectures.** Selected works. More details are in Appendix A.

Hierarchical design	**Decoupling via long skip**	**Decoupling via cross-attention**
		
ADM (Dhariwal and Nichol, 2021), SiD / SiD2 (Hoogeboom et al., 2023; 2025), PixelFlow (Chen et al., 2025b)	PixNerd (Wang et al., 2026a), DiP (Chen et al., 2026b), DeCo (Ma et al., 2026), PixelDiT (Yu et al., 2026)	RIN (Jabri et al., 2023), HyperDiT (He et al., 2026), DuSPiT (Bai et al., 2026)

Patch geometry and neural-network training. Natural-image patch geometry has long served as a prior in image processing and vision (Dabov et al., 2007; Peyré, 2009; Yu et al., 2012; Coates et al., 2011; Thiry et al., 2021; Brutzkus et al., 2022), rather than as a factor in how neural networks train. From the implicit-bias perspective, Liang et al. (2026b) characterize how *input-side* patch geometry affects gradient descent in two-layer patch-based ReLU networks. Inspired by this viewpoint, we study the complementary *output-side* patch geometry in modern diffusion Transformers and trace how it shapes hidden representations across depth.

Representations for generation. Diffusion models learn useful visual features (Xiang et al., 2023; Tang et al., 2023; Luo et al., 2023; Chen et al., 2025c; Stracke et al., 2025), though weaker than those of pretrained visual encoders (Yu et al., 2025), and stronger intermediate representations are closely associated with faster training and better generation (Yu et al., 2025; Leng et al., 2025; Wu et al., 2025b; Singh et al., 2026a). Our work suggests that one source of this representation-learning difficulty is *residual-stream burden*, which can create friction between noisy-target prediction and representation learning. The RAE family instead performs diffusion directly in pretrained representation spaces and converges much faster (Zheng et al., 2026; Singh et al., 2026b). Section 6 discusses how residual-stream burden bears on this advantage, and Appendix A gives further background.

3 Background and Problem Setup

Flow matching. We study the ODE formulation of flow-based models, where a neural network parameterizes a time-dependent velocity field (Lipman et al., 2023; Liu et al., 2023). Let $\boldsymbol{x} \sim p_{\text{data}}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$ be independent. We use the linear interpolation

$$\boldsymbol{x}_t = t\boldsymbol{x} + (1-t)\boldsymbol{\epsilon}, \qquad \boldsymbol{v} = \mathrm{d}\boldsymbol{x}_t/\,\mathrm{d}t = \boldsymbol{x} - \boldsymbol{\epsilon}, \qquad t \in [0,1]. \tag{1}$$

Thus $t = 0$ is noise and $t = 1$ is clean data. The network is trained with $\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\boldsymbol{x},\boldsymbol{\epsilon}} \|\boldsymbol{v}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{v}\|_2^2$, using the training time distribution on $(0,1)$. For $t < 1$, the target also satisfies $\boldsymbol{v} = (\boldsymbol{x} - \boldsymbol{x}_t)/(1-t)$. Under $\boldsymbol{v}$-prediction the network outputs $\boldsymbol{v}_\theta$ directly, and under $\boldsymbol{x}$-prediction it outputs the clean image $\boldsymbol{x}_\theta$, which is converted as $\boldsymbol{v}_\theta = (\boldsymbol{x}_\theta - \boldsymbol{x}_t)/(1-t)$ (Li and He, 2026). Sampling solves $\mathrm{d}\boldsymbol{x}_t/\,\mathrm{d}t = \boldsymbol{v}_\theta(\boldsymbol{x}_t, t)$ from Gaussian noise at $t = 0$ to data at $t = 1$, by default with a 50-step Heun solver following Li and He (2026). Class conditioning is omitted from the notation.

Residual stream. A residual network passes a running representation through depth, updating it recursively as $\boldsymbol{h}_{\ell+1} = \boldsymbol{h}_\ell + \mathcal{F}_\ell(\boldsymbol{h}_\ell, t)$ (He et al., 2016). Starting from the input embedding $\boldsymbol{h}_0$, each attention or MLP sublayer reads the current representation and adds its output to it. This sequence of representations forms the *residual stream*, in which information carried forward by the identity path also supplies the input to subsequent computation. In a plain Transformer, the width-C stream itself serves as each layer's computational *workspace*.

We now introduce the central notion of this paper.

Residual-stream burden is the requirement to preserve prediction-relevant input information through depth for the final readout.

A toy experiment to visualize the residual-stream burden. Following Li and He (2026, Section 3.3), the synthetic ground-truth data are a planar Swiss roll embedded in $\mathbb{R}^{512}$. The orthogonal embedding matrix is unknown to the model. We train toy flow-matching models to learn this distribution in $\mathbb{R}^{512}$, using a five-layer residual ReLU fully connected network (FCN) of width 256.

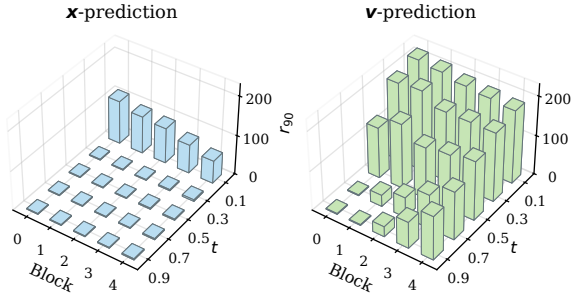


Figure 1: **Toy experiment.** Each bar gives the number of directions needed to explain 90% of the hidden-state variance. The hidden states under $\boldsymbol{x}$-prediction are low-dimensional, whereas those under $\boldsymbol{v}$-prediction are not, illustrating the burden.

The generation results are consistent with those reported by Li and He (2026): $\boldsymbol{x}$-prediction recovers the distribution and $\boldsymbol{v}$-prediction fails. Our focus is on the internal states. Figure 1 shows that the FCN with $\boldsymbol{x}$-prediction concentrates residual-state variance in a few directions except at the highest noise level, while $\boldsymbol{v}$-prediction needs over a hundred to explain 90% of the variance by the last block. Even when the input is nearly clean, the FCN with $\boldsymbol{v}$-prediction still spreads residual-state variance over many directions, so later layers receive broadly varying states. More details are in Appendix D.

This toy experiment makes concrete how the prediction target shapes internal representations in the residual stream. In this paper, we examine this insight with diffusion Transformers on real data.

4 Exploitable Low-Dimensionality in Patch Space

We first identify the exploitable low-dimensional structure in target patches (Section 4.1), then trace the way it shapes input filtering and the resulting intermediate representations (Section 4.2). Finally, we show that a similar low-dimensionality and representation-learning behavior emerge in the main stream of DiT blocks in recent decoupled designs in pixel space with $\boldsymbol{v}$-prediction (Section 4.3).

4.1 Spectral Concentration in Patch Space

A DiT processes (noisy) images patch by patch, so the relevant geometry is the *patch geometry*. It is therefore natural to ask whether clean images are low-dimensional in patch space. For clean 16×16 RGB patches in ImageNet 256^2, just eight of 768 principal directions explain 90% of the variance and preserve recognizable image structure (Figure 2), while the velocity target $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{\epsilon}$ requires 668 directions to reach the same fraction (Table 2 and Appendix B.7).



Figure 2: **A few patch directions preserve recognizable structure.** Each reconstruction keeps the indicated number of principal directions within every 16×16 patch. Eight directions preserve the shape of the mountain, and additional directions recover texture.

To test the role of spectral concentration, we whiten clean patches by an invertible linear transform that flattens the spectrum while mapping the clean-data manifold to another smoothly equivalent manifold. Training in the whitened space raises $\boldsymbol{x}$-prediction FID from 10.19 to 132.95, suggesting that spectral concentration in patch space may underlie the success of $\boldsymbol{x}$-prediction.

Prediction target Data space	Clean Pixel	Velocity Pixel	Clean Whitened
FID ↓	10.19	139.83	132.95
Effective rank	4.80	367.37	768
Stable rank	1.41	5.94	768
90% variance rank	8	668	692

Table 2: **Whitening degrades $\boldsymbol{x}$-prediction.**

4.2 How Patch Spectral Concentration Shapes Representations

Following the toy experiment (Figure 1), we now show how patch spectral concentration is reflected in the representations along the residual stream.

Spectral concentration induces selective patch filtering. The optimal velocity estimator is $\mathbb{E}[\boldsymbol{v} \mid \boldsymbol{x}_t, t] = (\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t, t] - \boldsymbol{x}_t)/(1-t)$. Under direct $\boldsymbol{v}$-prediction, the network must reproduce the noisy-input term, so its residual stream must carry corruption-dependent variation even along the many weak directions created by spectral concentration, where $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t, t]$ varies little. Under $\boldsymbol{x}$-prediction, the analytic long skip supplies this term, so the backbone only needs $\mathbb{E}[\boldsymbol{x} \mid \boldsymbol{x}_t, t]$, and the patch embedding, where the residual stream begins, can suppress these directions (Appendix B.4).

To examine this filtering, we measure the directional gain $g_i = \|\boldsymbol{W}_{\text{in}}\boldsymbol{u}_i\|^2$, which quantifies how strongly the embedding transmits variation along the clean-patch principal direction $\boldsymbol{u}_i$, and compare the eigenvectors of $\boldsymbol{W}_{\text{in}}^\top \boldsymbol{W}_{\text{in}}$ with these directions (Appendix B.5). Pixel $\boldsymbol{x}$-prediction assigns 61.5% of total gain to the leading 64 clean-patch directions and only 0.04% to the trailing 64, compared with 10.0% and 7.0% for $\boldsymbol{v}$-prediction (Figure 3).

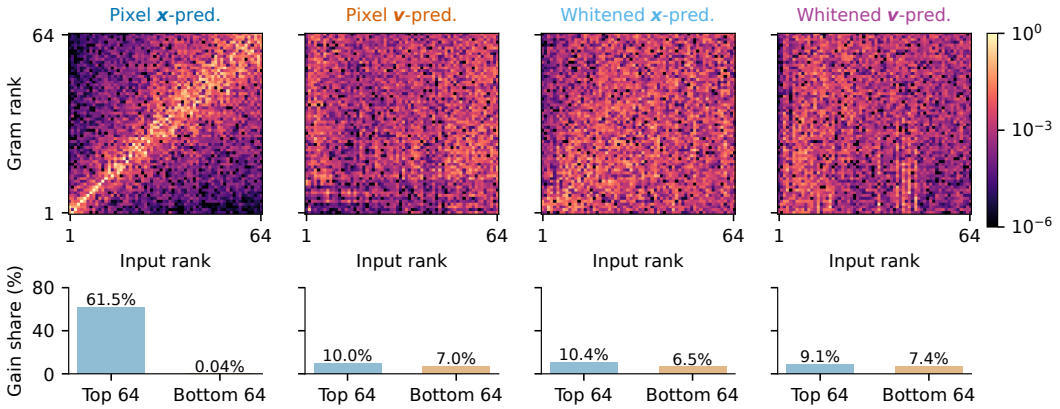


Figure 3: **Selective filtering depends on target and patch geometry.** Top: squared overlaps between the leading 64 input directions and embedding-Gram eigenvectors; a bright diagonal indicates alignment. Bottom: embedding gain on the leading and trailing 64 directions, normalized over all 768. Whitened axes retain the original clean-PCA order.

Noise burden and representation quality. For two noisy versions $\boldsymbol{x}_t$ and $\boldsymbol{x}'_t$ of the same clean image, $\boldsymbol{v}$-prediction must keep their noise-dependent differences distinguishable in the residual stream, whereas $\boldsymbol{x}$-prediction does not, since the long skip supplies $\boldsymbol{x}_t$. This gives the backbone greater freedom to learn noise-robust representations. Linear probes along the residual stream (Figure 4) show that pixel-space $\boldsymbol{x}$-prediction develops much stronger class features than $\boldsymbol{v}$-prediction, whereas both whitened models remain poorly class-readable, even under $\boldsymbol{x}$-prediction. Together with the embedding measurements, these results connect spectral concentration to selective noise filtering and stronger representations, whose quality is closely associated with generation quality (Yu et al., 2025).

4.3 Decoupled Designs Exploit Low-Dimensionality in the Main Stream

Decoupled architectures offer another way to exploit low-dimensional clean structure while predicting velocity. DeCo, PixelDiT and DiP supply noisy pixels directly to a decoder conditioned on the main Transformer's features (Table 1), allowing local noise-dependent variation to reach the output through a separate path. We call the features passed to the decoder the *semantic endpoint*, which reflects the residual-stream burden on the main backbone of DiT blocks. As expected, their spectra are also concentrated (see Figure 5). A low-dimensional interface thus emerges naturally through the end-to-end training of the decoupled Transformer designs.

As in a plain DiT with $\boldsymbol{x}$-prediction, this structure is reflected at the backbone's entrance and through its depth. Specifically, the patch embeddings align with leading clean-patch directions and suppress trailing directions (Figure 6), while intermediate states become much more class-readable than those of plain $\boldsymbol{v}$-prediction (Figure 4). These observations reconcile the decoder-based designs with

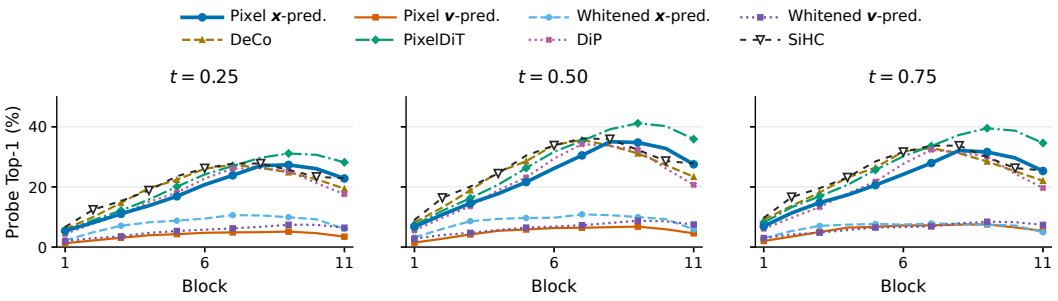


Figure 4: **Class readability follows the burden.** Linear-probe Top-1 accuracy along the residual stream for the plain DiT in pixel and whitened space, three decoupled $\boldsymbol{v}$-prediction models (Section 4.3) and the shared workspace of SiHC (Section 5.2). Pixel $\boldsymbol{x}$-prediction and the decoupled models are markedly more class-readable than pixel $\boldsymbol{v}$-prediction and both whitened models. Columns fix the clean coefficient t, and shading shows the standard deviation across noise draws.

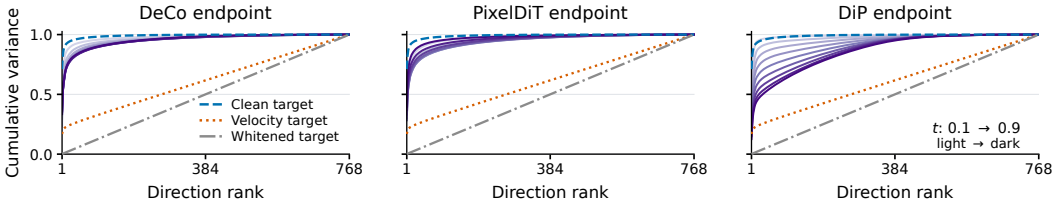


Figure 5: **Semantic endpoints are far more concentrated than the velocity target.** Target spectra provide a reference for the prediction demand. A faster rise means that fewer directions carry the variance. All three B-size models train successfully under $\boldsymbol{v}$-prediction, reaching FID 6.33–13.31 compared with 139.83 for the plain DiT (Table 20).

$\boldsymbol{x}$-prediction: both provide a separate path for noisy input, accompanied by selective filtering and stronger representations in the main stream. Even a zero-initialized scalar skip from the input to the output learns such a path under $\boldsymbol{v}$-prediction, reaching FID 9.99 at 200 epochs (Appendix B.8).

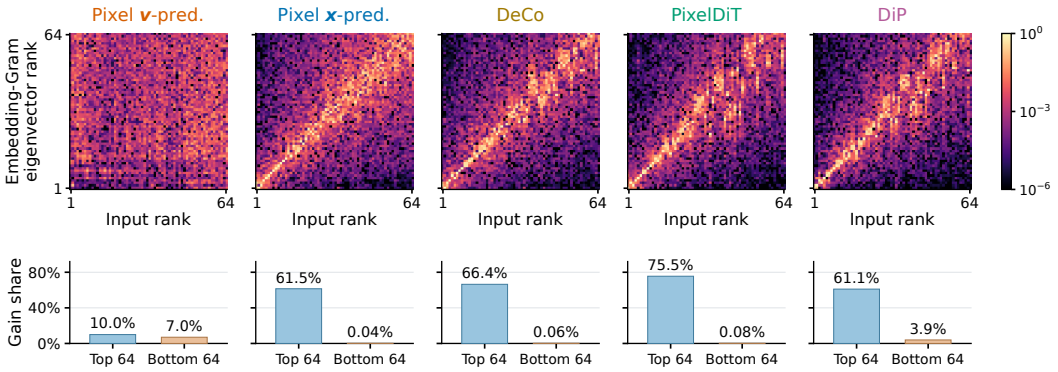
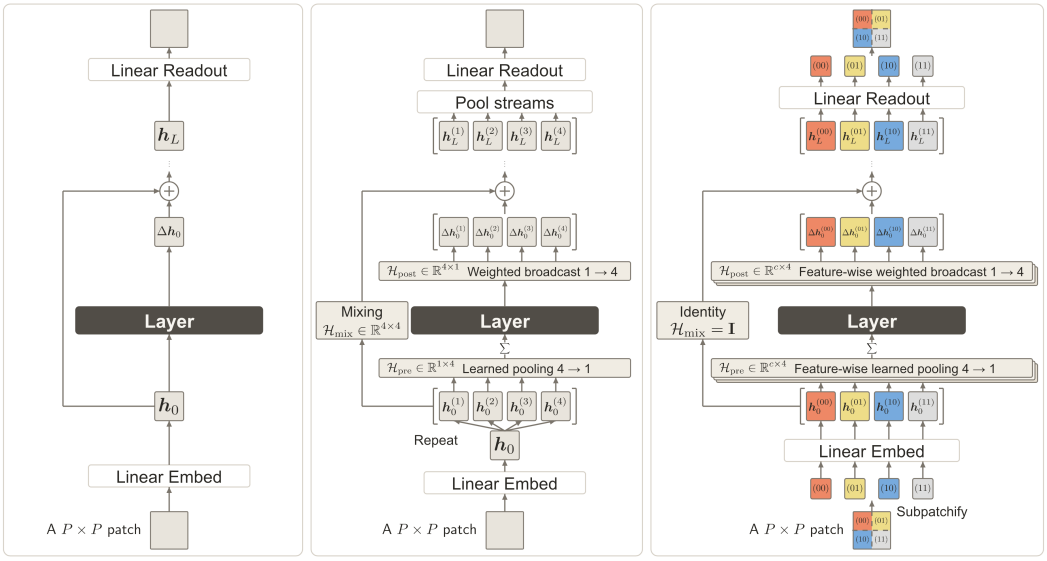


Figure 6: **Decoupled models filter their input like $\boldsymbol{x}$-prediction.** The top row shows squared overlaps between clean-patch principal directions and embedding-Gram eigenvectors, where a bright diagonal indicates alignment. The bottom row shows the share of input gain on the leading and trailing 64 clean-patch directions.

Together, these results connect patch geometry, prediction targets and architecture through residual-stream burden. We use the output target-patch covariance spectrum as a measurable proxy for *the size* of this burden: it describes how broadly target variation is distributed across directions. By this measure, the velocity target places a far larger burden than the clean target (r_{90} of 668 against 8). The architecture, in turn, determines how this burden is assigned to the main stream and separate paths: with a separate pixel path, the semantic endpoints of decoupled models need only 35 to 309 directions at $t = 0.9$ (Appendix B.7). When the burden instead stays on the main stream, as in plain $\boldsymbol{v}$-prediction, the stream itself may need more room. We next test whether expanding the residual stream helps accommodate this broader prediction demand.



(a) Standard residual connection (b) Hyper-Connections (c) Spatially indexed Hyper-Connections (ours)

Figure 7: **Persistent state and workspace in three residual designs,** read from bottom to top. (a) A plain residual state is also the layer workspace. (b) HC carries several states but reads one width-C workspace for each update. (c) SiHC assigns each state a local input and prediction, tracked by color, and connects the states to shared computation through feature-wise read/write maps.

5 Accommodating Residual-Stream Burden

Having shown that separate pixel paths can relieve the main backbone of carrying noise to the final output, we now ask whether expanding the *capacity of the persistent state* lets the backbone accommodate this requirement internally. We call the persistent state's width the *residual-stream bandwidth*. In a plain Transformer, increasing it also widens every attention and MLP computation, since the residual state itself serves as the workspace (Figure 7(a)). Hyper-Connections (HC) separate persistent state from workspace (Zhu et al., 2025a), allowing us to expand residual-stream bandwidth without widening those computations.

5.1 Expanding Residual-Stream Bandwidth with Hyper-Connections

HC replaces $\boldsymbol{h}_\ell$ with S persistent states $\boldsymbol{H}_\ell = [\boldsymbol{h}_\ell^{(1)}, \dots, \boldsymbol{h}_\ell^{(S)}]^\top \in \mathbb{R}^{S\times C}$, from which each update reads one width-C workspace before writing its result back (Figure 7(b)),

$$(\boldsymbol{h}_\ell^{\text{in}})^\top = \boldsymbol{\mathcal{H}}_{\text{pre}}^\ell \boldsymbol{H}_\ell, \qquad \boldsymbol{H}_{\ell+1} = \boldsymbol{\mathcal{H}}_{\text{mix}}^\ell \boldsymbol{H}_\ell + \boldsymbol{\mathcal{H}}_{\text{post}}^\ell (\Delta \boldsymbol{h}_\ell)^\top, \qquad \Delta \boldsymbol{h}_\ell = \mathcal{F}_\ell(\boldsymbol{h}_\ell^{\text{in}}, t). \tag{2}$$

The read map $\boldsymbol{\mathcal{H}}_{\text{pre}}^\ell \in \mathbb{R}^{1\times S}$ pools the states, the write map $\boldsymbol{\mathcal{H}}_{\text{post}}^\ell \in \mathbb{R}^{S\times 1}$ distributes the update, and $\boldsymbol{\mathcal{H}}_{\text{mix}}^\ell \in \mathbb{R}^{S\times S}$ mixes the carried states. The residual-stream bandwidth becomes SC while the workspace width stays C. In dynamic HC, these maps are computed for every token from its current states, so the connections adapt to the input (Zhu et al., 2025a). In particular, mHC (Xie et al., 2026), a dynamic variant that keeps $\boldsymbol{\mathcal{H}}_{\text{mix}}^\ell$ doubly stochastic for stable propagation through depth, has been adopted in open-source LLMs such as DeepSeek-V4 (DeepSeek-AI et al., 2026).

The broader target spectrum of $\boldsymbol{v}$-prediction suggests a larger benefit from added bandwidth than for $\boldsymbol{x}$-prediction (Section 4). We test this prediction in both pixel space and the DINOv2-B latent space used by RAE (Zheng et al., 2026) using four-stream mHC to expand the bandwidth at nearly unchanged parameters and GFLOPs (depth, token count and computational widths remain fixed). With this mHC intervention, $\boldsymbol{v}$-prediction improves substantially: its pixel-space FID falls from 139.8 to 25.4 and added bandwidth nearly closes the gap in DINOv2-B space.

Prediction	Plain FID ↓	mHC FID ↓
Pixel, $\boldsymbol{x}$	10.19	9.78
Pixel, $\boldsymbol{v}$	139.83	25.36
DINOv2-B, $\boldsymbol{x}$	5.50	3.93
DINOv2-B, $\boldsymbol{v}$	16.56	3.98

Table 3: **HC helps $\boldsymbol{v}$-prediction most.** All models are B-size and use Heun-50 with CFG 2.9. Each pixel patch or DINOv2-B token has 768 coordinates.

5.2 Spatially Indexed Hyper-Connections

Copied streams expand bandwidth but leave the division of the burden among streams to training. *Spatially Indexed Hyper-Connections* (SiHC) make this division explicit by assigning each persistent state the input and output of one subpatch, while attention/MLP layers read a shared workspace from all states (Figure 7(c)), keeping the number of workspace tokens unchanged.

(1) **Local states carry local predictions.** Each $P \times P$ computational patch is divided into $S = (P/b)^2$ subpatches. State s embeds only its noisy subpatch, $\boldsymbol{h}_0^{(s)} = \boldsymbol{W}_{\text{in}} \boldsymbol{p}_t^{(s)}$, and predicts its local velocity, $\widehat{\boldsymbol{v}}^{(s)} = \boldsymbol{W}_{\text{out}} \boldsymbol{h}_{\text{final}}^{(s)}$. This reduces each state's prediction responsibility from $3P^2$ to $3b^2$ pixel coordinates, while the S local predictions tile the full velocity patch.

(2) **Identity carry keeps local detail available.** With $\boldsymbol{\mathcal{H}}_{\text{mix}}^{\ell} = \boldsymbol{I}_S$, each state retains its own carry path independently of its contribution to the current workspace.

(3) **Static feature-wise access connects local states to shared computation.** Learned, input-independent maps $\boldsymbol{\mathcal{H}}_{\text{pre}}^{\ell}, \boldsymbol{\mathcal{H}}_{\text{post}}^{\ell} \in \mathbb{R}^{C \times S}$, shared across spatial cells, let each channel read its own combination of states and distribute its update with its own weights.

For channel c, the read and write around each attention or MLP update are

$$h_{\ell,c}^{\text{in}} = \textstyle\sum_s \boldsymbol{\mathcal{H}}_{\text{pre}}^{\ell}[c,s]\, h_{\ell,c}^{(s)}, \qquad h_{\ell+1,c}^{(s)} = h_{\ell,c}^{(s)} + \boldsymbol{\mathcal{H}}_{\text{post}}^{\ell}[c,s]\, \Delta h_{\ell,c}. \tag{3}$$

Read-back makes local information available to subsequent computation, and write-back distributes the resulting updates among local prediction states. Dynamic maps must be recomputed from the full S-stream state before every update, which requires materializing the wide state at every sublayer. With static maps and identity carry, reads and writes instead compose across a stage, allowing exact kernel fusion that materializes the wide state only at stage boundaries. This late materialization reduces the memory traffic and footprint of the expanded state, lowering peak training memory from 30.18 to 19.82 GiB for the sixteen-stream B model (Appendix G).

Ablation study. Table 4 tests these choices in sequence on a B-size backbone with 16×16 computational patches and direct $\boldsymbol{v}$-prediction. Assigning each of four streams its own 8×8 input and prediction region lowers FID from 25.4 to 13.2 at the same bandwidth, demonstrating the value of smaller local responsibilities. Static scalar maps with identity carry improve FID further, and channel-specific read/write maps refine access to the local states without input-dependent connections. Dividing the patch into sixteen 4×4 regions then expands bandwidth while reducing each state's prediction responsibility to 48 pixel coordinates, bringing FID to 8.07, within the range of decoupled B-size models (Table 20).

Table 4: **Each design choice improves $\boldsymbol{v}$-prediction.** Parameters and GFLOPs remain nearly unchanged across the progression.

Design	What changes	State region	Streams	FID ↓
Plain DiT		16×16	1	139.83
mHC, copied input	Four persistent states	16×16	4	25.36
mHC, spatial input	Local input and prediction per state	8×8	4	13.25
+ identity carry, scalar maps	Static maps without state mixing	8×8	4	11.50
+ feature-wise maps (SiHC)	Channel-specific read and write	8×8	4	10.10
+ 4×4 subpatches	More states, smaller regions	4×4	16	8.07

The sublayerwise SiHC workspace also develops much stronger class features than plain $\boldsymbol{v}$-prediction, tracking pixel-space $\boldsymbol{x}$-prediction through depth (Figure 4). Organizing persistent state and its access thus recovers class-readable features without a separate pixel decoder. Further noise-sensitivity measurements appear in Appendix C.4.

This design also scales beyond the B-size analysis (Appendix E.4). With REPA, SiHC-XL reaches FID 1.71 (Table 5), competitive with strong pixel-space models while retaining linear read/write maps in place of specialized decoders or cross-scale attention.

6 Discussion, Limitations and Future Work

The high-level view of this work is to treat a neural network not as a function from noisy input to target but as a system whose persistent state must serve both prediction and representation learning.

Table 5: **Pixel-space generation on ImageNet-256^2.** We compare with a selection of recent pixel-space models under a common sampling setting: all rows use Heun-50, and our evaluations follow JiT's FID protocol (Li and He, 2026). Some of these models achieve better results with their own sampling configurations, for example DeCo and PixelDiT. These results and further details are given in Appendix A.5. †Uses REPA; –: not reported.

Model	Prediction	Params (M)	GFLOPs	FID ↓	IS ↑
JiT-H/16 (Li and He, 2026)	$\boldsymbol{x}$	953	364	1.86	303.4
JiT-G/16 (Li and He, 2026)	$\boldsymbol{x}$	2000	767	1.82	292.6
PixelREPA-H/16 (Shin et al., 2026)	$\boldsymbol{x}$	953	364	1.81	317.2
PixNerd-XL/16† (Wang et al., 2026a)	$\boldsymbol{v}$	700	268	1.89	309.0
DiP-XL/16† (Chen et al., 2026b)	$\boldsymbol{v}$	631	255	1.75	273.0
DeCo-XL/16† (Ma et al., 2026)	$\boldsymbol{v}$	682	245	1.69	304.0
PixelDiT-XL† (Yu et al., 2026)	$\boldsymbol{v}$	797	311	1.69	292.0
AsymFlow-H/16† (Chen et al., 2026a)	$\boldsymbol{x} - \boldsymbol{P}\boldsymbol{\epsilon}$	953	363	1.57	–
SiHC-XL† (Ours)	$\boldsymbol{v}$	763	299	1.71	299.6

From this view, residual-stream burden explains a central difficulty in training diffusion models: predicting a noisy target requires the residual stream to keep the noise, and this induces friction with representation learning. Analyzing the patch space provides proxies for the size of the burden and shows why the issue becomes especially severe for high-dimensional data and large patches, where the noise spreads over far more directions. Through this lens, we reconcile recent progress in pixel-space diffusion, where changing the prediction parameterization and changing the architecture succeed for a common reason, namely relieving the main stream of the noise. Building on this understanding, we explore a complementary direction that accommodates the burden by expanding residual-stream bandwidth, with SiHC serving as an algorithmic probe.

Limitations. Our analysis shows that residual-stream burden shapes the learned representations, but it does not trace how this shaping unfolds during training, which is fundamentally difficult. Representations in deep networks emerge from gradient dynamics, in which representation learning and gradient geometry are coupled. The gradient of a hidden unit combines its inputs, weighted by backpropagated errors, and these inputs are themselves learned by earlier layers, so each representation update also changes the geometry of subsequent updates. This coupling prevents us from tracing how specific representations form, and it also leaves open how much bandwidth representation learning itself requires. Architecture and data geometry both act by reshaping the network parameters through these dynamics, and existing theoretical tools remain insufficient to characterize the resulting *implicit bias* precisely (Appendix A.4).

Future directions. Viewing residual-stream burden through its friction with representation learning raises several questions.

(1) *How does the implicit bias of gradient descent interact with friction?* Selective filtering, concentrated semantic endpoints and a learned long skip (Appendix B.8) all emerge without an explicit objective. We conjecture that when the architecture offers a path that carries the burden outside feature-building computation, gradient descent drifts toward solutions with less friction, consistent with a preference for simpler solutions. Testing this conjecture requires a way to characterize friction during training, for example as interference between the gradients that preserve the target and those that build features.

(2) *Can residual-stream burden benefit representation learning?* When the target is itself a pretrained representation, as in RAE (Zheng et al., 2026; Singh et al., 2026b), the burden consists of features that later layers can build on and may even facilitate representation learning, offering a possible reason for the fast convergence of RAE. This also invites studying *diffusability*, the question of which latent space suits diffusion models (Skorokhodov et al., 2025; Yao et al., 2025; Leng et al., 2025), from the perspective of neural network training.

(3) *When do generation and understanding reinforce each other?* The same training perspective extends to unified models of generation and visual understanding (Wu et al., 2025a; Hu et al., 2026; Wang et al., 2026b). Recent unified models that generate within a shared semantic latent space report improved alignment between the two tasks (Jin et al., 2026). Characterizing the burden that generation places on shared representations, and when it is compatible with understanding, may help answer this question.

AI Use Statement. Generative AI tools were used to assist with language editing and improve the readability of the manuscript, and to prepare preliminary versions of figures and visualizations. Codex and Claude Code also assisted with code development and debugging, including the implementation of kernel fusion and the selection of tiling configurations. All AI-assisted text, figures, and code were reviewed, revised as needed, and verified by the authors. The authors formulated the main research ideas, made the methodological and experimental decisions, and are responsible for the interpretation of results and the scientific conclusions. The authors take full responsibility for the final content of this work.

Reproducibility Statement. The supplementary materials include anonymous SiHC source code with model configurations, training and evaluation entry points, fused kernels, benchmark utilities, correctness tests and a toy-analysis suite. The appendices document the data transformations and spectral measurements (Appendix B), representation diagnostics (Appendices C and H), toy setup (Appendix D), training and sampling protocols (Appendix E), and the read/write implementation and fusion benchmarks (Appendices F and G).

REFERENCES

Yunpeng Bai, Yossi Gandelsman, and Michaël Gharbi. DuSPiT: Dual-branch sub-patch pixel diffusion transformer. *arXiv preprint arXiv:2607.18510*, 2026.

Alon Brutzkus, Amir Globerson, Eran Malach, Alon Regev Netser, and Shai Shalev-Schwartz. Efficient learning of CNNs using patch based features. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2336–2356. PMLR, 2022.

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

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *International Conference on Learning Representations*, 2025a.

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

Xinlei Chen, Zhuang Liu, Saining Xie, and Kaiming He. Deconstructing denoising diffusion models for self-supervised learning. In *International Conference on Learning Representations*, 2025c.

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

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223. PMLR, 2011.

Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

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 of *Proceedings of Machine Learning Research*, pages 9550–9575. PMLR, 2024.

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.

Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *International Conference on Learning Representations*, 2023.

Alexandru Damian, Jason D. Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Proceedings of the 35th Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 5413–5452, 2022.

DeepSeek-AI et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794, 2021.

Funing Fu, Tenghui Wang, Guanyu Zhou, Junyong Cen, and Qichao Zhu. JLT: Clean-latent prediction in latent diffusion transformers. *arXiv preprint arXiv:2605.27102*, 2026.

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1832–1841, 2018a.

Suriya Gunasekar, Jason D. Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018b.

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 *Computer Vision – ECCV 2026*, volume 17060 of *Lecture Notes in Computer Science*, pages 425–442. Springer, 2026.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

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

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion: 1.5 FID on ImageNet512 with pixel-space diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18062–18071, 2025.

Taihang Hu, Mengting Chen, Jinsong Lan, Xiaoyong Zhu, Kaifu Zhang, Ming-Ming Cheng, Bo Zheng, and Yaxing Wang. ORION: Decoupling and alignment for unified autoregressive understanding and generation. In *International Conference on Learning Representations*, 2026.

Allan Jabri, David J. Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In *International Conference on Machine Learning*, pages 14569–14589, 2023.

Dengyang Jiang, Ruoyi Du, Zhennan Chen, Dongyang Liu, Zanyi Wang, Mingzhe Zheng, Xiangpeng Yang, Huanqia Cai, Aiming Hao, Yuming Jiang, Peng Gao, Harry Yang, and Steven Hoi. An empirical study of training pixel-space text-to-image diffusion models. *arXiv preprint arXiv:2608.16887*, 2026.

Jiachun Jin, Zetong Zhou, Xiao Yang, Hao Zhang, Pengfei Liu, Jun Zhu, and Zhijie Deng. LatentUM: Unleashing the potential of interleaved cross-modal reasoning via a latent-space unified model. *arXiv preprint arXiv:2604.02097*, 2026.

Qing Jin and Chaoyang Wang. Revisiting diffusion model predictions through dimensionality. *arXiv preprint arXiv:2601.21419*, 2026.

Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. REPA-E: Unlocking VAE for end-to-end tuning of latent diffusion transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18262–18272, 2025.

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

Tongtong Liang, Dan Qiao, Yu-Xiang Wang, and Rahul Parhi. Stable minima of ReLU neural networks suffer from the curse of dimensionality: The neural shattering phenomenon. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Tongtong Liang, Alexander Cloninger, Rahul Parhi, and Yu-Xiang Wang. Generalization below the edge of stability: The role of data geometry. In *International Conference on Learning Representations*, 2026a.

Tongtong Liang, Esha Singh, Rahul Parhi, Alexander Cloninger, and Yu-Xiang Wang. Does sparse connectivity improve generalization? convolutional networks below the edge of stability. *arXiv preprint arXiv:2603.04807*, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Grace Luo, Lisa Dunlap, Dong Huk Park, Aleksander Holynski, and Trevor Darrell. Diffusion hyperfeatures: Searching through time and space for semantic correspondence. In *Advances in Neural Information Processing Systems*, volume 36, pages 47500–47510, 2023.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *Computer Vision – ECCV 2024*, volume 15135 of *Lecture Notes in Computer Science*, pages 23–40. Springer, 2024.

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

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: Dimension-free bounds and kernel limit. In *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 2388–2464, 2019.

Rotem Mulayoff, Tomer Michaeli, and Daniel Soudry. The implicit bias of minima stability: A view from function space. In *Advances in Neural Information Processing Systems*, volume 34, pages 17749–17761, 2021.

Mor Shpigel Nacson, Rotem Mulayoff, Greg Ongie, Tomer Michaeli, and Daniel Soudry. The implicit bias of minima stability in multivariate shallow ReLU networks. In *International Conference on Learning Representations*, 2023.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Lukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1559–1571. Association for Computational Linguistics, 2022.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. 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 Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

Gabriel Peyré. Manifold models for signals and images. *Computer Vision and Image Understanding*, 113(2):249–260, 2009.

Dan Qiao, Kaiqi Zhang, Esha Singh, Daniel Soudry, and Yu-Xiang Wang. Stable minima cannot overfit in univariate ReLU networks: Generalization by large step sizes. In *Advances in Neural Information Processing Systems*, volume 37, pages 94163–94208, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

Seed, Baisheng Li, Banggu Wu, Bole Ma, Bowen Xiao, Chaoyi Zhang, Cheng Li, Chengyi Wang, Chengyin Xu, Chi Zhang, et al. Virtual width networks. *arXiv preprint arXiv:2511.11238*, 2025.

Jaeyo Shin, Jiwook Kim, and Hyunjung Shim. Representation alignment for just image transformers is not easier than you think. In *Computer Vision – ECCV 2026*, volume 17013 of *Lecture Notes in Computer Science*, pages 603–620. Springer, 2026.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seung Eun Yi, Michael Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *Transactions on Machine Learning Research*, 2026.

Jaskirat Singh, Xingjian Leng, Zongze Wu, Liang Zheng, Richard Zhang, Eli Shechtman, and Saining Xie. What matters for representation alignment: Global information or spatial structure? In *International Conference on Learning Representations*, 2026a.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved baselines with representation autoencoders. *arXiv preprint arXiv:2605.18324*, 2026b.

Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 55876–55905. PMLR, 2025.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. Technical Report CRCV-TR-12-01, Center for Research in Computer Vision, University of Central Florida, 2012.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70):1–57, 2018.

Nick Stracke, Stefan Andreas Baumann, Kolja Bauer, Frank Fundel, and Björn Ommer. CleanDIFT: Diffusion features without noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 117–127, 2025.

Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. In *Advances in Neural Information Processing Systems*, volume 36, pages 1363–1389, 2023.

Tencent Hy Team. Hy4 preview model card, 2026. URL `https://huggingface.co/tencent/Hy4-preview`. Accessed September 14, 2026.

Louis Thiry, Michael Arbel, Eugene Belilovsky, and Edouard Oyallon. The unreasonable effectiveness of patches in deep convolutional kernels methods. In *International Conference on Learning Representations*, 2021.

Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. JetFormer: An autoregressive generative model of raw images and text. In *International Conference on Learning Representations*, 2025.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. PixNerd: Pixel neural field diffusion. In *International Conference on Learning Representations*, 2026a.

Shuai Wang, Liang Li, Yang Chen, Ruopeng Gao, Yao Teng, and Limin Wang. UniDDT: Unifying multimodal understanding and generation with decoupled diffusion transformer. In *Computer Vision – ECCV 2026*, volume 17042 of *Lecture Notes in Computer Science*, pages 294–313. Springer, 2026b.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. DDT: Decoupled diffusion transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 40633–40642, 2026c.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, and Ping Luo. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12966–12977, 2025a.

Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Jian Yang, Ming-Ming Cheng, and Xiang Li. Representation entanglement for generation: Training diffusion transformers is much easier than you think. In *Advances in Neural Information Processing Systems*, volume 38, pages 7714–7743, 2025b.

Lei Wu and Weijie J. Su. The implicit regularization of dynamical stability in stochastic gradient descent. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 37656–37684, 2023.

Lei Wu, Chao Ma, and Weinan E. How SGD selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Weilai Xiang, Hongyu Yang, Di Huang, and Yunhong Wang. Denoising diffusion autoencoders are unified self-supervised learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15802–15812, 2023.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Mingyu Xu, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mHC: Manifold-constrained hyper-connections. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

Chao Xu, Maohua Li, Qirui Li, Yixuan Xu, Yanke Zhou, Yunhe Li, Cuifeng Shen, Hanlin Tang, Kan Liu, Tao Lan, Lin Qu, and Shao-Qun Zhang. Rethinking cross-layer information routing in diffusion transformers. *arXiv preprint arXiv:2605.20708*, 2026.

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

Guoshen Yu, Guillermo Sapiro, and Stéphane Mallat. Solving inverse problems with piecewise linear estimators: From Gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2012.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.

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

Z.ai. GLM-5.3-Flash model card, 2026. URL `https://huggingface.co/zai-org/GLM-5.3-Flash`. Accessed September 9, 2026.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

Xiangdong Zhang, Xiaohan Qin, Sunan Zou, Tuo Dai, Xiaoming Shi, Huaijin Wu, Yebin Yang, Zhuo Xia, Shaofeng Zhang, Lin Yao, Yuliang Liu, Yu Cheng, and Junchi Yan. xHC: Expanded hyper-connections. *arXiv preprint arXiv:2607.14530*, 2026.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, 2026.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, 2025a.

Defa Zhu, Hongzhi Huang, Jundong Zhou, Zihao Huang, Yutao Zeng, Banggu Wu, Qiyang Min, and Xun Zhou. Frac-connections: Fractional extension of hyper-connections. *arXiv preprint arXiv:2503.14125*, 2025b.

A Additional Related Work

A.1 Diffusion Models and Prediction Targets

Noise, clean-data and velocity prediction describe the same diffusion process through interconvertible outputs and time-dependent loss weights (Ho et al., 2020; Salimans and Ho, 2022; Lipman et al., 2023; Liu et al., 2023), but they can assign different prediction responsibilities to the network. For example, clean prediction under a velocity-space objective supplies the noisy-input term through an analytic skip, changing what the residual stream must preserve (Li and He, 2026). Recent studies examine the resulting learning differences through prediction dimensionality, covariance anisotropy and pixel-space training (Jin and Wang, 2026; Fu et al., 2026; Jiang et al., 2026). We connect these differences to learned input filtering and the representations available to subsequent layers (Sections 4.2 and 4.3).

A.2 Latent Diffusion and Pixel-Space Modeling

Latent diffusion reduces denoising cost through autoencoder compression (Rombach et al., 2022), enabling scalable Transformer generators such as DiT and SiT (Peebles and Xie, 2023; Ma et al., 2024). This efficiency also makes generation depend on the representation supplied by the autoencoder: reconstruction accuracy alone does not determine how easily diffusion can learn the latent distribution (Yao et al., 2025; Skorokhodov et al., 2025). Aligning latents with visual features, jointly tuning the autoencoder and denoiser, and generating directly in pretrained representation spaces address this interaction from complementary directions (Yao et al., 2025; Leng et al., 2025; Zheng et al., 2026).

Why pixel space? Pixel-space modeling pursues a self-contained image synthesis pipeline in which the generative objective shapes the full path to image detail, without a separately pretrained image tokenizer. End-to-end diffusion and autoregressive modeling pursue this integration (Hoogeboom et al., 2023; Tschannen et al., 2025). Learning the image synthesis path jointly also avoids fixing the compression–reconstruction trade-off before training the generator (Rombach et al., 2022; Chen et al., 2025a). Recent pixel-space architectures make this integration practical by separating coarse computation from detailed prediction (Chen et al., 2026b; Yu et al., 2026). Large patches reduce attention cost but assign many pixel coordinates to each residual stream, making this setting useful for studying how prediction requirements interact with representation learning.

A.3 Architecture Design in Diffusion Models

Diffusion architectures distribute global modeling and detailed prediction through hierarchy or decoupled interfaces. U-Net, hourglass and staged designs allocate computation across resolutions (Dhariwal and Nichol, 2021; Hoogeboom et al., 2025; Nawrot et al., 2022; Crowson et al., 2024; Chen et al., 2025b; Guo et al., 2026), whereas decoupled designs assign detailed prediction to a head conditioned on backbone features, in both latent and pixel space (Wang et al., 2026c; Zheng et al., 2026; Chen et al., 2026b; Ma et al., 2026; Yu et al., 2026; Wang et al., 2026a). Analytic input skips induced by prediction reparameterization provide another way to supply local information outside the main stream (Li and He, 2026; Chen et al., 2026a). The common design principle is to preserve access to detail without requiring all of it to pass through the shared computation.

The interface determines how that detail remains accessible: decoder-based designs use final backbone features, while cross-attention connects data or fine-grained states to latent or coarse computation (Jabri et al., 2023; He et al., 2026; Bai et al., 2026). SiHC uses persistent local states with feature-wise linear access, keeping both state and workspace resolutions fixed through depth (Table 6). Our analysis connects these choices to prediction requirements: even at fixed granularity, the target and patch geometry change the burden on the residual stream (Section 4.3).

Table 6: **Prediction interfaces in pixel-space models.** Analytic skips, conditioned decoders and cross-attention provide different paths from noisy inputs to predictions. SiHC connects persistent local states to shared computation through linear read/write maps.

Model	Noisy-input path	Connection to backbone	Prediction computation
Analytic output paths			
JiT (Li and He, 2026)	Noisy input enters both the patch embedding and an analytic output skip.	Final patch features predict the clean image.	Linear patch head, then analytic conversion to velocity.
AsymFlow (Chen et al., 2026a)	An analytic input path recovers velocity outside a chosen noise subspace.	A JiT backbone predicts full-dimensional data minus low-rank noise.	Patch head with projection-based analytic velocity recovery.
Decoders conditioned on final backbone features			
DiP (Chen et al., 2026b)	Noisy RGB patches enter local U-Nets.	Context tokens enter U-Net bottlenecks.	Patch-local convolutions with down/up blocks and skips.
DeCo (Ma et al., 2026)	Noisy RGB and positions form pixel queries.	Upsampled coarse features supply AdaLN-zero conditioning.	Pixel-wise residual MLPs without attention.
PixelDiT (Yu et al., 2026)	Noisy pixels enter narrow tokens.	Patch tokens supply per-pixel AdaLN parameters.	Global attention on compressed tokens, then expansion and pixel MLPs.
PixNerd (Wang et al., 2026a)	Noisy RGB and coordinates query local neural fields.	Patch tokens predict normalized MLP weights.	Neural-field MLP and output projection per pixel.
Cross-attention interfaces across depth			
RIN (Jabri et al., 2023)	Noisy data are embedded as interface tokens.	Cross-attention reads into latent tokens and writes back to interface tokens.	Self-attention in latent tokens, with output decoded from the interface.
HyperDiT (He et al., 2026)	Separate embeddings initialize coarse semantic and fine-grained streams.	Fine tokens query semantic anchors from multiple backbone depths via cross-attention.	Hyper-Connector blocks refine the fine stream with scale-aware positional encoding.
DuSPiT (Bai et al., 2026)	Parallel patch and subpatch embeddings initialize two branches.	Each subpatch token attends only to its corresponding patch token, using intermediate patch-branch features.	Cross-attention and feed-forward updates in the pixel branch, followed by a clean-image head.
Linear access to persistent local states			
SiHC (ours)	Subpatch states persist with identity carry.	Feature-wise linear reads/writes connect states to shared sublayers.	Local output projections, with attention/MLPs in the shared workspace.

A.4 Implicit Bias of Gradient Descent

The filtering patterns in Sections 4.2 and 4.3 can be viewed as an expression of implicit bias: training selects patch directions without an explicit regularizer prescribing that selection, and the directions it retains change with the target and output path. This perspective follows the question raised by Zhang et al. (2017): when the same network can fit random labels yet generalize on natural data, which properties of training favor useful solutions? For diffusion, the question extends inside the network, because matching the output target leaves open how the model distributes information and computation across its layers.

Existing theory makes parts of this selection process precise under tractable assumptions. Analyses of linear models characterize how the loss, parameterization and optimization geometry induce margin or norm preferences (Soudry et al., 2018; Gunasekar et al., 2018a;b). For nonlinear networks, mean-field descriptions track how features evolve during training, while two-layer analyses establish advantages from learning task-relevant directions (Mei et al., 2019; Damian et al., 2022).

A complementary approach studies which solutions can remain stable under finite-step training, using the step size to constrain the solutions without tracking the full optimization trajectory (Wu et al., 2018; Wu and Su, 2023). The empirical edge-of-stability phenomenon and its self-stabilization mechanism connect this perspective to observed training dynamics (Cohen et al., 2021; Damian et al., 2023). For shallow ReLU networks, stability constraints translate into function-space regularity (Mulayoff et al., 2021; Nacson et al., 2023) and generalization guarantees in univariate regression (Qiao et al., 2024). Their extension to multivariate inputs reveals why dimension and data geometry matter (Liang et al., 2025; 2026a), while sparse connectivity changes the relevant geometry from whole inputs to local patches (Liang et al., 2026b). This line of work makes explicit how architecture and data jointly determine the regularization supplied by training.

The central difficulty is that the geometry driving optimization is itself learned. Even in a two-layer ReLU network, the gradient of a hidden neuron's input weights combines input vectors weighted by prediction errors, output weights and activation gates (Liang et al., 2026b). As those weights and gates change, so do the directions favored by subsequent updates. In a deep network, the inputs to later layers also move as earlier representations change, while backpropagated gradients depend on the computation built above them. Representation learning therefore changes the optimization problem experienced by each layer throughout training. Knowing which information an optimal predictor needs is insufficient to determine which internal representations gradient descent will construct to carry it.

Our experiments give this coupled problem a concrete empirical form. Changing the target alters which patch directions the embedding retains, and changing the output path can recover selective filtering under velocity prediction. Expanding residual-stream bandwidth changes how much information can remain available alongside the shared computation. A theory of these observations would need to explain how target-induced gradients select input directions and how their retention affects feature learning through depth. Residual-stream burden organizes these questions around the information that must remain accessible, linking the observed filtering and bandwidth effects to the optimization challenge discussed in Section 6.

A.5 Generation Performance and Training Context

Table 7 extends the main pixel-space comparison to latent models. REPA use is marked in the model names.[1] PixNerd's listed FID 1.93 uses 320 training epochs and Euler-100, while DiP's 1.79 uses 600 epochs and Euler-100 (Wang et al., 2026a; Chen et al., 2026b).

Sampling configurations. Table 5 uses Heun-50 throughout. We evaluate the released PixNerd, DiP and PixelDiT models, together with SiHC, using JiT's FID protocol on 50,000 samples. Their FID/IS pairs are $1.89/309.0$, $1.75/273.0$, $1.69/292.0$ and $1.71/299.6$, respectively. JiT, PixelREPA and DeCo use their reported Heun-50 results. AsymFlow's reported FID 1.57 uses ADM evaluation, and its corresponding IS is not reported (Chen et al., 2026a). Its asymmetric target is written as $\boldsymbol{x} - \boldsymbol{P}\boldsymbol{\epsilon}$ in our time convention, where $\boldsymbol{P}$ is a rank-eight patch projector.

Table 7 retains the broader comparison under each source's published sampling configuration. DeCo's FID 1.62 uses Euler-250 at 800 epochs (Ma et al., 2026). PixelDiT's FID 1.61 uses FlowDPMSolver-100 at 320 epochs, with guidance scale 2.75 over $[0.10, 0.90]$ (Yu et al., 2026). RAEv2 uses its DINOv3-L $K = 7$ configuration with REPA guidance (Singh et al., 2026b). GFLOPs measure one denoiser forward pass, while total sampling compute also depends on the solver, step count and classifier-free guidance.

[1] DiP's released XL configuration enables its REPA trainer: `https://github.com/NJU-PCALab/DiP/blob/main/configs_c2i/dip_xl.yaml`.

Table 7: **Generation quality and model cost on ImageNet-256^2.** Reported results span pixel space and latent spaces based on VAEs or RAEs. GFLOPs count one denoiser forward pass; training and sampling schedules follow each source. REPA is indicated explicitly for the pixel models that use it. –: not reported. *Computed from the published or released configuration (Appendix E.6).

Model	Params (M)	GFLOPs	FID ↓	IS ↑
Latent diffusion with VAE				
DiT-XL/2 (Peebles and Xie, 2023)	675	238	2.27	278.2
SiT-XL/2 (Ma et al., 2024)	675	238	2.06	277.5
SiT-XL/2 + REPA (Yu et al., 2025)	675	238	1.42	305.7
LightningDiT-XL/2 (Yao et al., 2025)	675	238	1.35	295.3
DDT-XL/2 (Wang et al., 2026c)	675	238	1.26	310.6
Latent diffusion with RAE				
RAE, DiT-XL (Zheng et al., 2026)	676	238	1.41	309.4
RAE, DiT$^{\mathrm{DH}}$-XL (Zheng et al., 2026)	839	323	1.13	262.6
RAEv2, DiT$^{\mathrm{DH}}$-XL (Singh et al., 2026b)	875*	475*	1.06	255.3
Pixel-space diffusion				
PixelFlow-XL/4 (Chen et al., 2025b)	677	5818	1.98	282.1
PixNerd-XL/16 + REPA (Wang et al., 2026a)	700	268	1.93	298.0
JiT-H/16 (Li and He, 2026)	953	364	1.86	303.4
JiT-G/16 (Li and He, 2026)	2000	767	1.82	292.6
PixelREPA-H/16 (Shin et al., 2026)	953	364	1.81	317.2
DiP-XL/16 + REPA (Chen et al., 2026b)	631	255	1.79	281.9
DeCo-XL/16 + REPA (Ma et al., 2026)	682	245	1.62	301.0
PixelDiT-XL + REPA (Yu et al., 2026)	797	311	1.61	292.7
HyperDiT-XL + REPA (He et al., 2026)	676	551*	1.63	304.2
AsymFlow-H/16 + REPA (Chen et al., 2026a)	953	363	1.57	–
SiHC-XL + REPA	763	299	1.71	299.6

B Patch Spectrum and Embedding Details

B.1 Definitions

For the input embedding, stack the noisy image patches as rows:

$$\mathbf{H}^0 = \mathbf{P}_t \boldsymbol{W}_{\mathrm{in}}^\top + \mathbf{B}, \qquad \mathbf{P}_t = \mathrm{Patchify}(\boldsymbol{x}_t), \tag{4}$$

Here $\boldsymbol{W}_{\mathrm{in}} \in \mathbb{R}^{C\times d}$ maps each patch vector to a width-C state, and $\mathbf{B}$ collects additive bias and positional terms. The centered clean-patch covariance is $\boldsymbol{\Sigma}_p = \mathrm{Cov}(\boldsymbol{p})$, with eigendecomposition

$$\boldsymbol{\Sigma}_p = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top, \qquad \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_d), \qquad \lambda_1 \geq \cdots \geq \lambda_d. \tag{5}$$

For covariance eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$, let $p_i = \lambda_i / \sum_j \lambda_j$. We use

$$\mathrm{ER} = \exp\left(-\sum_i p_i \log p_i\right), \qquad \mathrm{SR} = \frac{\sum_i \lambda_i}{\lambda_1}, \qquad r_\alpha = \min\left\{k : \frac{\sum_{i=1}^k \lambda_i}{\sum_i \lambda_i} \geq \alpha\right\}. \tag{6}$$

These statistics characterize the variance spectrum. Under independent isotropic corruption, the noisy patch covariance is

$$\mathrm{Cov}(\boldsymbol{p}_t) = t^2 \boldsymbol{\Sigma}_p + (1-t)^2 \boldsymbol{I}. \tag{7}$$

For the covariance of clean 16×16 patches, clean ER is 4.796 and the leading principal component explains 71.13% of variance. At $t = 0.5$, noisy-patch ER is 367.371 and $r_{90} = 668$. The raw velocity-target covariance and its rank are derived in Appendix B.7.

B.2 Patch spectra and additional visual measurements

The covariance and reconstruction basis use 100k ImageNet training images and 16×16 RGB patches. Figure 8 shows how corruption flattens the patch spectrum, and Figure 9 compares the clean and velocity target spectra in pixel and whitened space.

Remark 1 (JiT's bottleneck embedding). Li and He (2026) use a low-rank linear patch embedding in their "Just image Transformers" (JiT). JiT-B/64 successfully generates from 64×64 RGB patches with 12,288 coordinates using a bottleneck width of 128. Our learned-filtering results support this design: the fixed rank encourages the embedding to suppress noise-dominated directions before they enter the residual stream.

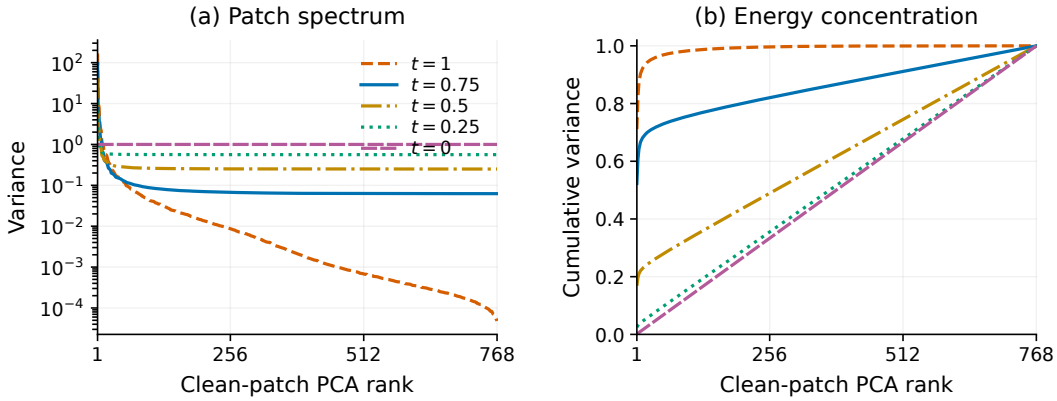


Figure 8: **Corruption spreads patch variance across directions.** Left: variance along clean-PCA directions. Right: cumulative variance. Decreasing the clean coefficient t flattens the spectrum and slows the cumulative rise, so more directions are needed to explain the same fraction of patch variance.

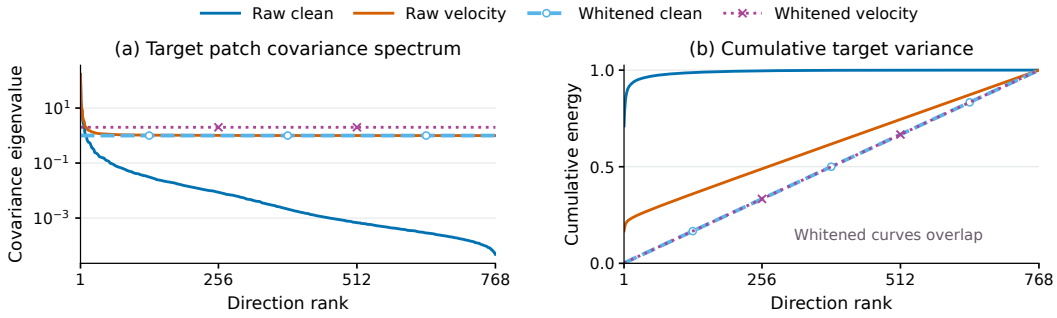


Figure 9: **Whitening broadens both prediction targets.** Left: target covariance eigenvalues. Right: cumulative variance. The whitened spectra assign equal variance to every direction, so their normalized cumulative curves coincide. Raw clean patches concentrate variance in a small leading subspace.

Figure 10 complements the alignment and head/tail views in the main text, and Figure 11 extends the reconstruction in Figure 2 to five images.

B.3 Spectral concentration in video tubelets

Spectral concentration is also present in natural video. We sample one RGB tubelet of size $4 \times 16 \times 16$ (time $\times$ height $\times$ width) from each of 5,050 UCF101 videos, with 50 videos per class (Soomro et al., 2012), taken from $16 \times 256 \times 256$ clips. Each tubelet has 3,072 coordinates with RGB values in $[-1, 1]$, and the covariance is centered by the global sample mean. Only nine of the 3,072 principal directions explain 90% of the variance (Figure 12), compared with eight of 768 for the 16×16 image patches in Section 4.1. Local spatiotemporal patches of natural video thus concentrate their variance in a few directions, as image patches do.

B.4 Best-affine prediction and directional gain

The conditional expectations in Section 4.2 describe optimal predictions under squared loss. Restricting to affine maps gives an explicit second-moment calculation of the variation predictable from the noisy input.

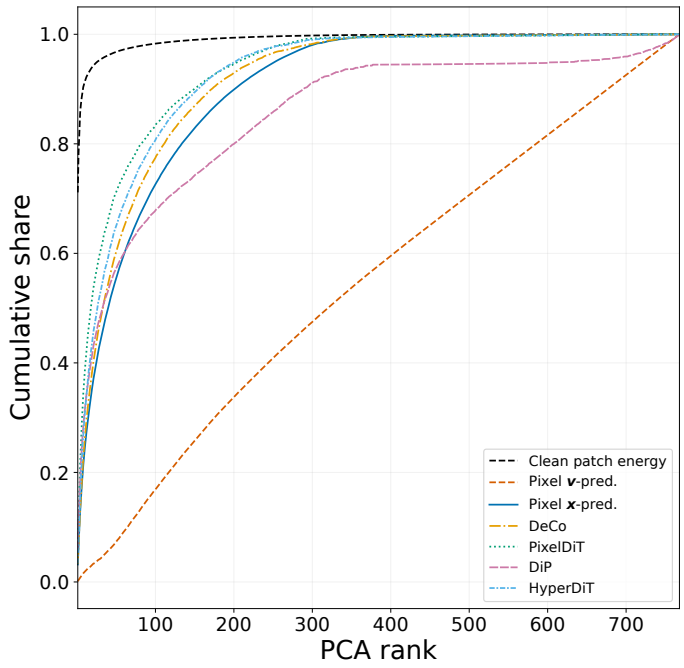


Figure 10: **Cumulative gain shows which patch directions enter the stream.** Each curve sums embedding gain in clean-PCA order. A faster rise means that more gain is assigned to the leading clean directions.

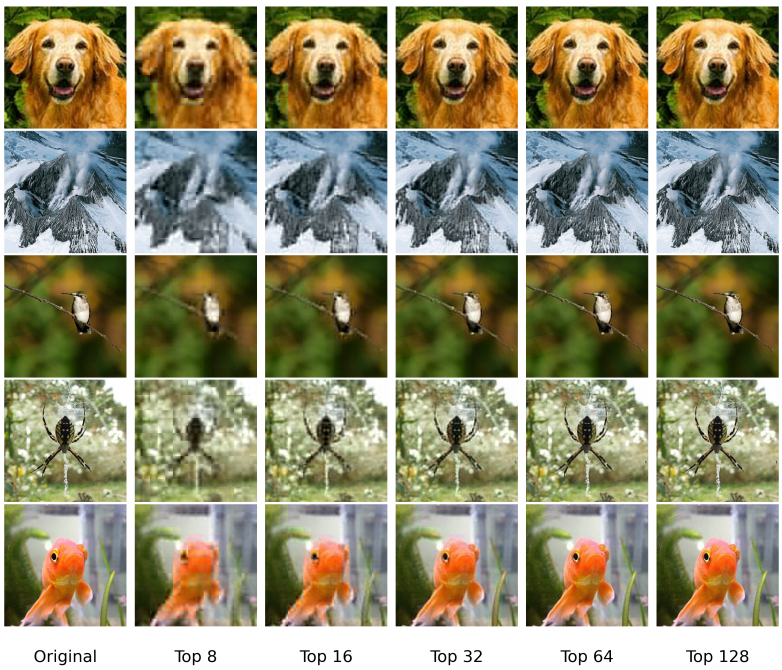
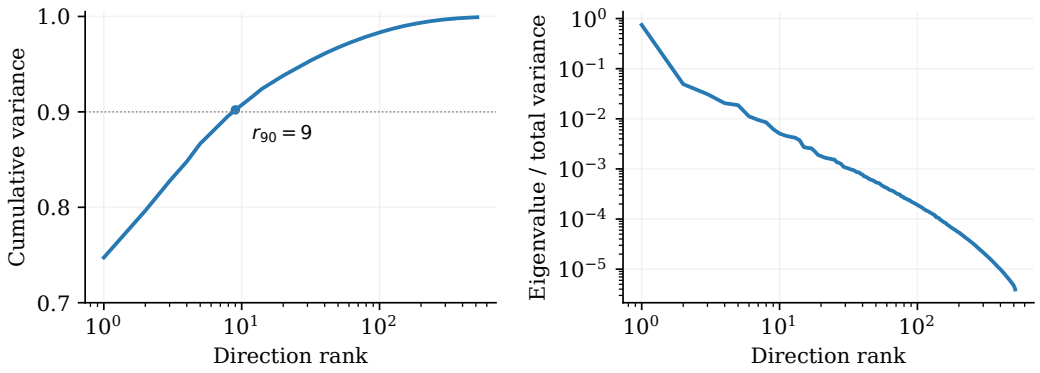


Figure 11: **Leading patch directions preserve recognizable structure across images.** Each row is one image. Columns show the original and reconstructions retaining 8, 16, 32, 64 and 128 principal directions within each 16×16 patch. More directions recover fine texture.



Tubelet	Dimension	r_{90}	r_{95}	r_{99}
UCF101, $4 \times 16 \times 16$	3,072	9	29	151

Figure 12: **Video tubelets have concentrated spectra.** Left: cumulative variance, with the marker at r_{90}. Right: covariance eigenvalues normalized by total variance.

For a centered clean patch $\boldsymbol{p}$ and independent noise $\boldsymbol{n} \sim \mathcal{N}(0, \boldsymbol{I})$, define the noisy patch $\boldsymbol{p}_t = t\boldsymbol{p} + \sigma\boldsymbol{n}$ with $\sigma = 1 - t$ and patch velocity $\boldsymbol{v}_p = \boldsymbol{p} - \boldsymbol{n}$. Write $\boldsymbol{\Sigma}_{tt} = \mathrm{Cov}(\boldsymbol{p}_t)$ and $\boldsymbol{\Sigma}_{vt} = \mathrm{Cov}(\boldsymbol{v}_p, \boldsymbol{p}_t)$. Their minimum-MSE affine map satisfies

$$\boldsymbol{A}_v = \boldsymbol{\Sigma}_{vt}\boldsymbol{\Sigma}_{tt}^{-1}, \qquad \mathrm{Cov}(\widehat{\boldsymbol{v}}_p) = \boldsymbol{\Sigma}_{vt}\boldsymbol{\Sigma}_{tt}^{-1}\boldsymbol{\Sigma}_{tv}, \qquad \widehat{\boldsymbol{v}}_p = \boldsymbol{A}_v\boldsymbol{p}_t. \tag{8}$$

For fixed $t < 1$, $\sigma = 1 - t > 0$. Independence of clean patches and unit Gaussian noise gives

$$\boldsymbol{\Sigma}_{tt} = t^2\boldsymbol{\Sigma}_p + \sigma^2\boldsymbol{I}, \qquad \boldsymbol{\Sigma}_{vt} = t\boldsymbol{\Sigma}_p - \sigma\boldsymbol{I}. \tag{9}$$

In the clean covariance eigenbasis, the covariance of the optimal affine velocity prediction therefore has entries

$$q_i^v(t) = \frac{(t\lambda_i - \sigma)^2}{t^2\lambda_i + \sigma^2}. \tag{10}$$

The corresponding affine clean prediction has covariance entries

$$q_i^p(t) = \frac{t^2\lambda_i^2}{t^2\lambda_i + \sigma^2}. \tag{11}$$

For fixed $t < 1$, these quantities approach 1 and 0, respectively, as λ_i approaches zero. A weak clean direction can thus contribute predictably to velocity even when it contributes almost nothing to the clean estimate. These are optimal affine predictors determined by the second moments. Table 8 uses the same ImageNet 16×16 patch spectrum for both targets, sorting each derived spectrum independently before computing r_{90}.

Table 8: **Affine clean prediction remains concentrated across noise levels.** Each entry gives the number of directions explaining 90% of predicted variance. Compare clean and velocity at each t.

Paper t	0.1	0.3	0.5	0.7	0.75	0.9
Affine clean r_{90}	1	2	3	5	5	7
Affine velocity r_{90}	674	656	638	603	588	488

The corresponding affine coefficients are $a_i^x = t\lambda_i/(t^2\lambda_i + \sigma^2)$ and $a_i^v = (t\lambda_i - \sigma)/(t^2\lambda_i + \sigma^2)$. For fixed $t < 1$, as $\lambda_i \to 0$, these coefficients tend to zero and $-1/\sigma$, respectively. The learned embedding gain is measured separately as described in Appendix B.5.

For the whitened covariance, $\lambda_i = 1$ in every direction, giving affine clean-prediction variance $t^2/(t^2 + (1-t)^2)$ in each coordinate. For comparison with these affine predictions, Figure 13 reports the raw target and learned embedding-Gram spectra.

B.5 Embedding metrics

For a clean-patch principal direction $\boldsymbol{u}_i$, the embedding's squared directional gain is

$$g_i = \|\boldsymbol{W}_{\mathrm{in}}\boldsymbol{u}_i\|_2^2. \tag{12}$$

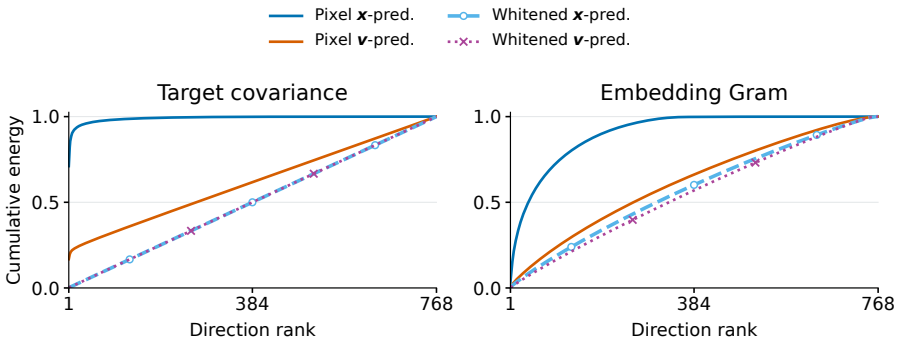


Figure 13: **Target and embedding spectra across coordinates.** Cumulative target covariance (left) and embedding-Gram eigenvalues (right), each sorted independently and normalized by its sum. Pixel $\boldsymbol{x}$-prediction has the most concentrated target and embedding spectra; whitening removes this concentration.

We normalize clean energy and embedding gain across these directions as

$$p_i = \frac{\lambda_i}{\sum_j \lambda_j}, \qquad \bar{g}_i = \frac{g_i}{\sum_j g_j}. \tag{13}$$

The normalized gain $\bar{g}_i$ measures the fraction of total embedding gain assigned to direction i, while p_i gives that direction's fraction of clean-patch variance.

The input-side Gram matrix $\boldsymbol{G} = \boldsymbol{W}_{\text{in}}^\top \boldsymbol{W}_{\text{in}}$ gives the same directional gain as $g_i = \boldsymbol{u}_i^\top \boldsymbol{G} \boldsymbol{u}_i$, and the top-$k$ gain share is $\sum_{i \le k} \bar{g}_i$. Subspace alignment is $\|\boldsymbol{U}_k^\top \boldsymbol{V}_k\|_F^2 / k$, where $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ contain the leading k eigenvectors of the clean covariance and embedding Gram matrix, respectively.

Table 9: Embedding gain assigned to the leading and trailing 256 clean-PCA directions, as a percentage of total gain over all 768 directions. Pixel $\boldsymbol{x}$ is the plain DiT with $\boldsymbol{x}$-prediction, and all other models use $\boldsymbol{v}$-prediction.

Gain share (%)	Pixel, $\boldsymbol{v}$	Pixel, $\boldsymbol{x}$	DeCo	PixelDiT	DiP	HyperDiT
Top 256	41.7	95.3	96.9	98.0	86.5	97.9
Bottom 256	28.0	0.2	0.2	0.3	5.4	0.3

Table 10: **Embeddings under x-prediction and decoupled designs align with the leading clean-patch subspace.** Higher mean squared cosine indicates stronger overlap. Columns expand the compared subspaces from $k = 16$ to $k = 256$.

Model	$k = 16$	$k = 32$	$k = 64$	$k = 128$	$k = 256$
Pixel, $\boldsymbol{x}$	0.903	0.925	0.921	0.909	0.949
Pixel, $\boldsymbol{v}$	0.076	0.095	0.268	0.530	0.652
DeCo-B	0.948	0.946	0.930	0.927	0.929
PixelDiT-B	0.920	0.877	0.885	0.891	0.935
DiP-B	0.924	0.929	0.884	0.726	0.770
HyperDiT-B	0.925	0.867	0.874	0.891	0.912

B.6 Target spectra in DINO space

In DINO space, the clean target is the normalized DINOv2-B latent used by RAE, one 768-dimensional feature per token. We estimate its covariance from 100,000 class-balanced ImageNet training images, using all 256 spatial tokens per image (25.6 million observations). Features use the encoder's final LayerNorm followed by RAE's released position-dependent normalization, without additional per-token RMS normalization. The covariance is centered globally over the pooled tokens.

Writing $\boldsymbol{\Sigma}_{\text{D}}$ for this measured clean covariance, independent unit-variance Gaussian corruption gives

$$\text{Cov}(\boldsymbol{v}) = \boldsymbol{\Sigma}_{\text{D}} + \boldsymbol{I}, \qquad \lambda_i^{\boldsymbol{v}} = \lambda_i^{\boldsymbol{x}} + 1. \tag{14}$$

Figure 14 plots the measured clean spectrum and the velocity spectrum derived from this identity. The clean DINO target already spreads its variance across many directions: $r_{90} = 563$ and $\text{ER} = 579.6$.

The velocity target is broader still, with $r_{90} = 651$ and ER $= 708.3$ (Table 11). Unlike pixel patches, whose r_{90} rises from 8 to 668, DINO features have broad clean variation before corruption is added. DINO and pixel velocity targets thus have similar r_{90}, while their final-boundary linear-probe accuracies differ markedly (Appendix H).

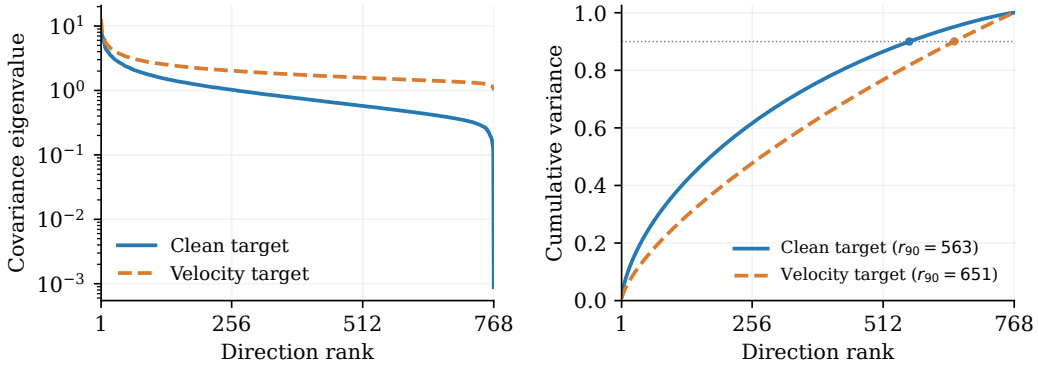


Figure 14: **DINO targets are broad under both prediction choices.** Left: covariance eigenvalues of RAE-normalized DINOv2-B tokens and the corresponding velocity target. Right: cumulative variance, with markers at r_{90}. Adding unit Gaussian variance broadens an already distributed clean spectrum.

Table 11: **DINO target ranks.** Statistics use the 768-dimensional spectra in Figure 14; definitions follow Appendix B.

Target	ER	SR	r_{90}	r_{95}	r_{99}
DINO, $\boldsymbol{x}$	579.6	66.84	563	649	735
DINO, $\boldsymbol{v}$	708.3	121.31	651	708	755

B.7 Embedding spectra and main-stream output interfaces

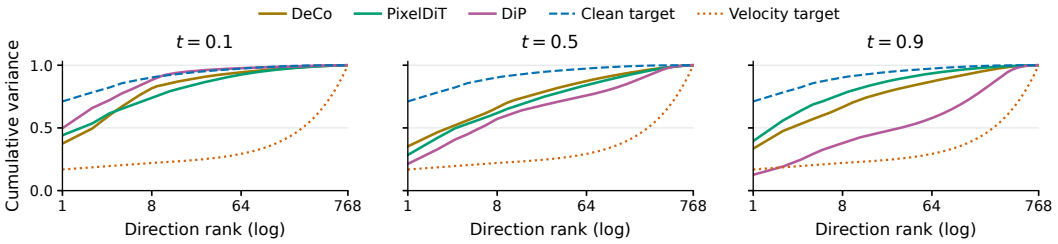


Figure 15: **Semantic endpoints concentrate variance relative to the velocity target.** Panels fix $t = 0.1, 0.5, 0.9$. Solid curves show the three semantic endpoints, with clean and velocity target spectra as dashed and dotted references. Direction rank uses a logarithmic axis to make the leading subspace visible.

Figure 16 reports embedding spectra, directional gains and main-stream output spectra. For decoupled models, the output measurement is taken at the interface that conditions the pixel decoder.

Endpoint measurements use raw epoch-200 B-size checkpoints and the registered 100k ImageNet training images, with true-class conditioning and one deterministic Gaussian draw per image shared across models and times. We record the 768-dimensional features passed from the main Transformer to the detail module, excluding decoder activations, at $t = 0.1, 0.2, \ldots, 0.9$. At $t = 0.9$, r_{90} is 97 for DeCo-B, 35 for PixelDiT-B, and 309 for DiP-B. The raw velocity target has covariance $\boldsymbol{\Sigma}_p + \boldsymbol{I}$ and a time-independent $r_{90} = 668$, whereas noisy input patches have covariance $t^2\boldsymbol{\Sigma}_p + (1-t)^2\boldsymbol{I}$ and a time-dependent spectrum. Figures 5 and 15 use target spectra as references, with the latter selecting three noise levels and a logarithmic rank axis.

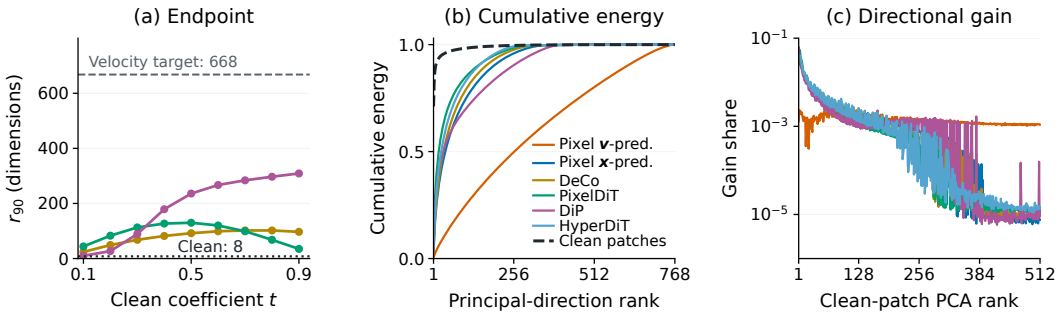


Figure 16: **Decoupled v-prediction models concentrate both main-stream interfaces.** (a) Lower r_{90} means fewer directions explain 90% of output-feature variance. The dashed lines mark raw-target references. (b) Faster-rising curves indicate a more concentrated embedding spectrum. (c) Gain along clean-PCA directions reveals which input variations enter the stream: pixel v-prediction retains a broader range than x-prediction and the decoupled models.

Raw velocity-target baseline. The plotted reference is the covariance of the raw patch target $v_p = p - n$. Independence gives $\mathrm{Cov}(v_p) = \Sigma_p + I$, so its eigenvalues are $\lambda_i + 1$. The leading 668 directions explain 90% of raw-target variance, giving $r_{90} = 668$.

Table 12: **Patch covariance and endpoint measurement protocols.** Each covariance treats image-position pairs as observations.

Setting	Value
Covariance fitting set	100,000 ImageNet training images, 100 per class
Selection / flip seed	20260806 / 20260807
Preprocessing	ADM center crop to 256^2, RGB in $[-1, 1]$; one deterministic horizontal-flip draw per selected image
Patch coordinates	Non-overlapping 16×16 RGB; channel, patch row, patch column order
Observations	$100{,}000 \times 256 = 25.6$M patches of dimension 768
Centering	One global 768-dimensional mean over all patches
Numerics	FP32 batch cross-products, FP64 merging and eigendecomposition; TF32 disabled
Whitening	All 768 PCA coordinates, divided by their fitted standard deviations
Endpoint weights	Raw epoch 200, step 250,200
Endpoint conditioning	True ImageNet class; one shared deterministic noise draw per image
Endpoint times / features	$t = 0.1, 0.2, \ldots, 0.9$; 768-dimensional main-stream output to the detail module
Embedding measurement	Eigenvalues/eigenvectors of $W_{\mathrm{in}}^\top W_{\mathrm{in}}$ from raw weights

B.8 A LEARNED LONG SKIP FOR VELOCITY PREDICTION

A learned scalar long skip provides a simple instance of decoupled prediction. We augment JiT-B/16 with a direct noisy-input path,

$$\widehat{v} = \alpha(t)x_t + \beta(t)f_\theta(x_t, t, y), \tag{15}$$

with $x_t = tx + (1-t)\epsilon$, as in the main text. A two-layer time-conditioned MLP produces the two scalar coefficients, initialized to $\alpha = 0$ and $\beta = 1$. The backbone uses a full-rank patch embedding, without context tokens or REPA, and the training objective remains v-prediction.

Figure 17 shows that the normalized coefficient $-(1-t)\alpha(t)$ approaches one at low and intermediate t within two epochs and across most of the time grid by epoch 40. Thus the learned input coefficient approaches $-1/(1-t)$, the input coefficient of the clean-to-velocity conversion $\widehat{v} = (\widehat{x} - x_t)/(1-t)$. At 200 epochs, FID is 9.99 compared with 139.83 for the plain v-prediction backbone under the same recipe, and continued training reaches 6.70 at 400 epochs.

For comparison, k-Diff reports that its learned prediction parameter moves toward clean prediction in pixel space (Jin and Wang, 2026). In our experiment, the skip starts at zero and the velocity loss acts on the sum of two independently weighted branches.

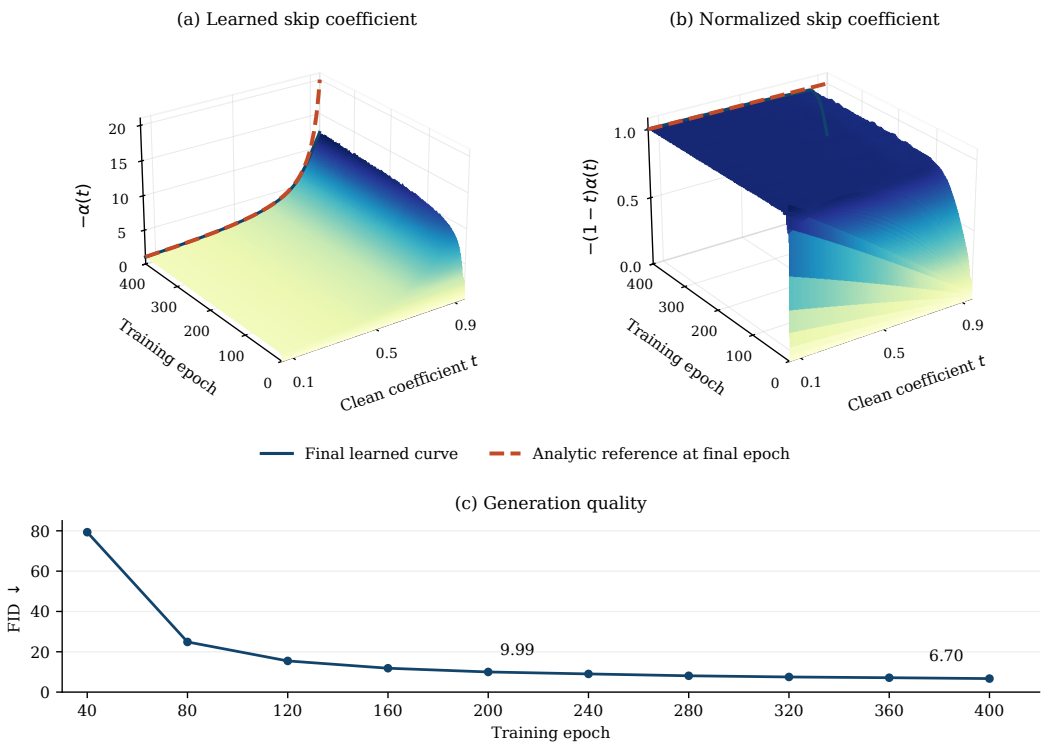


Figure 17: **The learned input skip approaches the clean-to-velocity coefficient during training.** Both surfaces show the clean coefficient t, training epoch and the learned skip coefficient. (a) $-\alpha(t)$, with the analytic reference $1/(1-t)$ drawn at the final recorded epoch. (b) $-(1-t)\alpha(t)$, with the corresponding reference at one. Solid blue lines show the final learned curves; dashed orange lines show the references. The surfaces use 246 recorded training snapshots, with denser sampling early in training and shape-preserving interpolation between the 20 measured time points. (c) FID at all ten evaluated checkpoints, using 50,000 samples, EMA weights and Heun-50 sampling; markers denote measurements.

B.9 Invertible patch whitening

Whitening tests the role of patch spectral concentration by flattening clean-patch variance while preserving manifold structure (Section 4.1). An invertible linear map takes a data manifold to one of the same intrinsic dimension; training the same model in this space raises clean-prediction FID from 10.19 to 132.95.

Because a DiT receives images through patch embeddings, we apply whitening to each patch, following the emphasis on architecture-dependent input geometry in Liang et al. (2026b). For centered patches $\boldsymbol{p}$, PCA rows $\boldsymbol{B}$ and scales $\boldsymbol{D} = \mathrm{diag}(\sqrt{\lambda_i})$ define $\boldsymbol{z} = \boldsymbol{D}^{-1}\boldsymbol{B}\boldsymbol{p}$ (Table 12). Applied to every patch, this is an invertible linear map of the whole image, and it sets every eigenvalue of the fitted clean-patch covariance to one. Flow matching then runs in this space as it does in pixel space, and generated samples are mapped back by $\boldsymbol{B}^\top \boldsymbol{D}\boldsymbol{z} + \boldsymbol{\mu}$ before FID evaluation.

Both targets use the direct JiT-B/16 backbone, the training settings in Table 17 and the sampling protocol in Table 19. The whitened velocity target has effective rank 768, stable rank 768 and 90% variance rank 692, and reaches FID 212.54. Table 13 gives the evaluation history.

Embedding measurements use raw epoch-200 weights, flattened to 768×768 in channel–row–column order, with FP64 singular-value decomposition. For singular values s_i, Gram energy is s_i^2, effective rank is $\exp(-\sum_i p_i \log p_i)$ with $p_i = s_i^2 / \sum_j s_j^2$, and r_{90} counts directions containing 90% of that energy. In native whitened coordinates, clean/velocity Gram effective ranks are $738.07/747.69$ and r_{90} values are $647/656$. The raw-pixel clean/velocity controls have effective ranks $175.06/696.24$ and r_{90} values $192/616$. Figure 13 compares native-coordinate weight spectra, and Figure 3 shows

Table 13: **Training history in whitened space.** Lower FID and higher IS are better.

Epoch	Clean FID	Clean IS	Velocity FID	Velocity IS
40	298.46	2.60	238.82	2.85
80	166.89	5.82	243.77	2.29
120	148.95	7.64	204.79	2.55
160	140.39	8.61	209.37	3.01
200	132.95	9.54	212.54	3.35

alignment and directional gain. Its heatmaps show squared overlaps between the leading 64 input-reference directions and the leading 64 eigenvectors of $\boldsymbol{W}_{\text{in}}^{\top}\boldsymbol{W}_{\text{in}}$. Raw-pixel references are clean-patch covariance eigenvectors. In whitened coordinates, we retain the coordinate axes defined by the rows of $\boldsymbol{B}$, ordered by the original patch variance, so the reference basis is $\boldsymbol{I}$.

C Normalized Representation Sensitivity

We measure how a normalized representation changes when the same clean image receives a new noise draw. Table 14 specifies the shared protocol for pixel, whitened and decoupled models.

C.1 Observation points and protocol

Table 14: **Normalized sensitivity protocol.** The spatial token grid is preserved when computing variance.

Setting	Value
Weights	Raw epoch 200 / step 250,200
Images	2,048 ImageNet training images, uniform sampling without replacement
Corruption	128 Gaussian draws per image at each $t \in \{0.25, 0.50, 0.75\}$
Pairing	Identical image indices and deterministic Gaussian draws across models/times
Input / conditioning	Pixel or whitened training coordinates (Appendix B.9); null class 1000; no image flips
Observation	Attention/MLP input before normalization and adaptive modulation
Readout	Per-token RMS, $\epsilon = 10^{-6}$; keep all 256 spatial positions
Observed width	768; $D = 256 \times 768 = 196{,}608$ coordinates
Depth	12 main-stream blocks; 8 for HyperDiT
Token selection	Spatial main-stream tokens; context and register tokens omitted
Numerics	BF16 forward, FP32 normalization, FP64 variance accumulation

The top row of Figure 18 selects the MLP input of each block. Its lower row uses next-attention inputs after blocks 1–11, with the probe fitting and spatial pooling described in Appendix H. All measurements use raw epoch-200 weights and precede native normalization and modulation. SiHC uses the workspace read from its persistent states in both rows. Figure 19 retains both attention and MLP observations in execution order. SiHC workspace measurements and the main SiHC progression in Table 4 use the sublayerwise implementation, whose attention and MLP have separate read/write maps.

C.2 Normalized noise variation and its image-dependent counterpart

Let $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\epsilon}; t)$ denote the normalized layer input for image $\boldsymbol{x}$ and noise $\boldsymbol{\epsilon}$, with $\boldsymbol{\mu}(\boldsymbol{x}; t) = \mathbb{E}_{\boldsymbol{\epsilon}}\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\epsilon}; t)$. Variation between images and variation from resampled noise are

$$B(t) = \mathbb{E}_{\boldsymbol{x}}\|\boldsymbol{\mu}(\boldsymbol{x}; t) - \mathbb{E}_{\boldsymbol{x}'}\boldsymbol{\mu}(\boldsymbol{x}'; t)\|_2^2, \qquad W(t) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon}}\|\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{\epsilon}; t) - \boldsymbol{\mu}(\boldsymbol{x}; t)\|_2^2. \tag{16}$$

The representation signal-to-noise ratio B/W measures between-image variation relative to corruption variation. The signal here consists of all image-dependent differences.

Let $\boldsymbol{r}_{ijk} \in \mathbb{R}^C$ be spatial token k for image i and noise draw j at a fixed observation point and time. We apply a separate, non-affine RMS normalization,

$$\widetilde{\boldsymbol{r}}_{ijk} = \frac{\boldsymbol{r}_{ijk}}{\sqrt{\|\boldsymbol{r}_{ijk}\|_2^2/C + 10^{-6}}}, \tag{17}$$

then concatenate the spatial tokens conceptually into $\boldsymbol{h}_{ij} \in \mathbb{R}^D$. Spatial locations remain separate throughout the variance computation. For $N = 2048$ images and $K = 128$ noises, define the image

means $\bar{\boldsymbol{h}}_i = K^{-1}\sum_j \boldsymbol{h}_{ij}$ and their grand mean $\bar{\boldsymbol{h}} = N^{-1}\sum_i \bar{\boldsymbol{h}}_i$. The within-image covariance trace and the plotted sensitivity are

$$\widehat{W} = \frac{1}{N(K-1)} \sum_{i=1}^{N}\sum_{j=1}^{K} \|\boldsymbol{h}_{ij} - \bar{\boldsymbol{h}}_i\|_2^2, \qquad \widehat{S} = \frac{\widehat{W}}{D}. \tag{18}$$

Thus S is the mean coordinate variance induced by changing the noise while holding the image fixed. At the population level, it is also half the mean squared distance per coordinate between two independently corrupted versions of the same image. Larger S means greater noise sensitivity of the normalized representation.

Small within-image variation is useful only if the representation still distinguishes images. We therefore retain the between-image covariance trace and its finite-K correction,

$$\widehat{B}_{\text{raw}} = \frac{1}{N-1} \sum_{i=1}^{N} \|\bar{\boldsymbol{h}}_i - \bar{\boldsymbol{h}}\|_2^2, \tag{19}$$

$$\widehat{B} = \widehat{B}_{\text{raw}} - \frac{\widehat{W}}{K}, \qquad \widehat{R} = \frac{\widehat{B}}{\widehat{W}}. \tag{20}$$

The correction removes noise variation left in the finite-sample image means, isolating image-dependent variation in the representation. B/D and R accompany S in Figure 19, allowing low sensitivity to be distinguished from a representation that varies little with either images or noise.

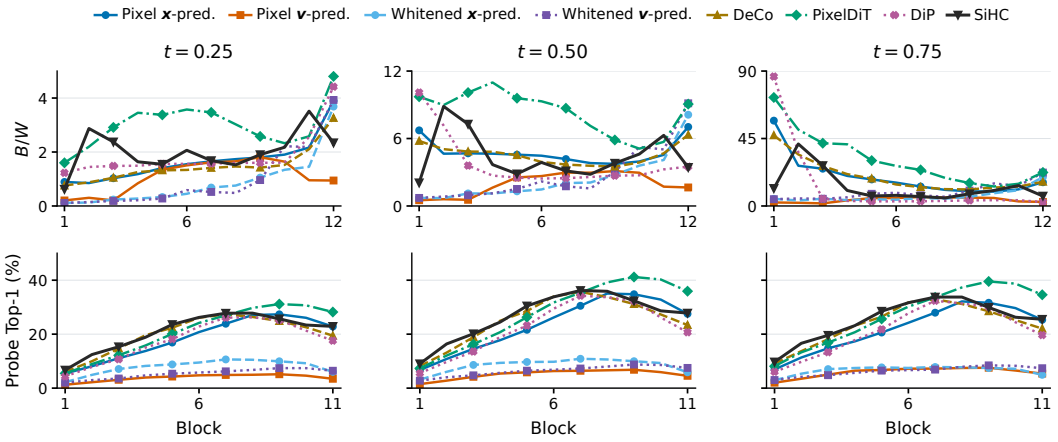


Figure 18: **Image-to-noise ratio and class readability through depth.** Top: variation between images relative to variation from noise (B/W). Bottom: class readability measured by linear-probe Top-1 accuracy. Curves cover the plain DiT in pixel and whitened space, the decoupled models and SiHC. Columns fix the clean coefficient t.

C.3 What changes through depth

Pixel $\boldsymbol{v}$-prediction reduces noise sensitivity through much of the backbone, followed by a late increase at the MLP inputs. At $t = 0.50$, S rises from 0.03948 at block 10 to 0.08516 at block 12, while pixel $\boldsymbol{x}$-prediction falls from 0.04348 to 0.01484. The velocity rebound occurs at all three measured times. Its noise variation grows faster than its between-image variation over these final blocks, so B/W falls (Figure 19).

The decoupled models show different depth profiles. DeCo and PixelDiT have lower final MLP-input sensitivity than pixel $\boldsymbol{v}$-prediction at every measured time. PixelDiT also shows a clear decrease at its last attention update. DiP is less sensitive than pixel $\boldsymbol{v}$-prediction at the final measured point but rises late at $t = 0.75$, while HyperDiT retains appreciable late sensitivity in its shorter semantic stream. Table 15 summarizes the depth averages.

Whitened models have high late-layer B/W ratios while their linear-probe accuracies remain low (Figure 18). Here B measures all image-dependent variation, whereas the probe measures class readability.

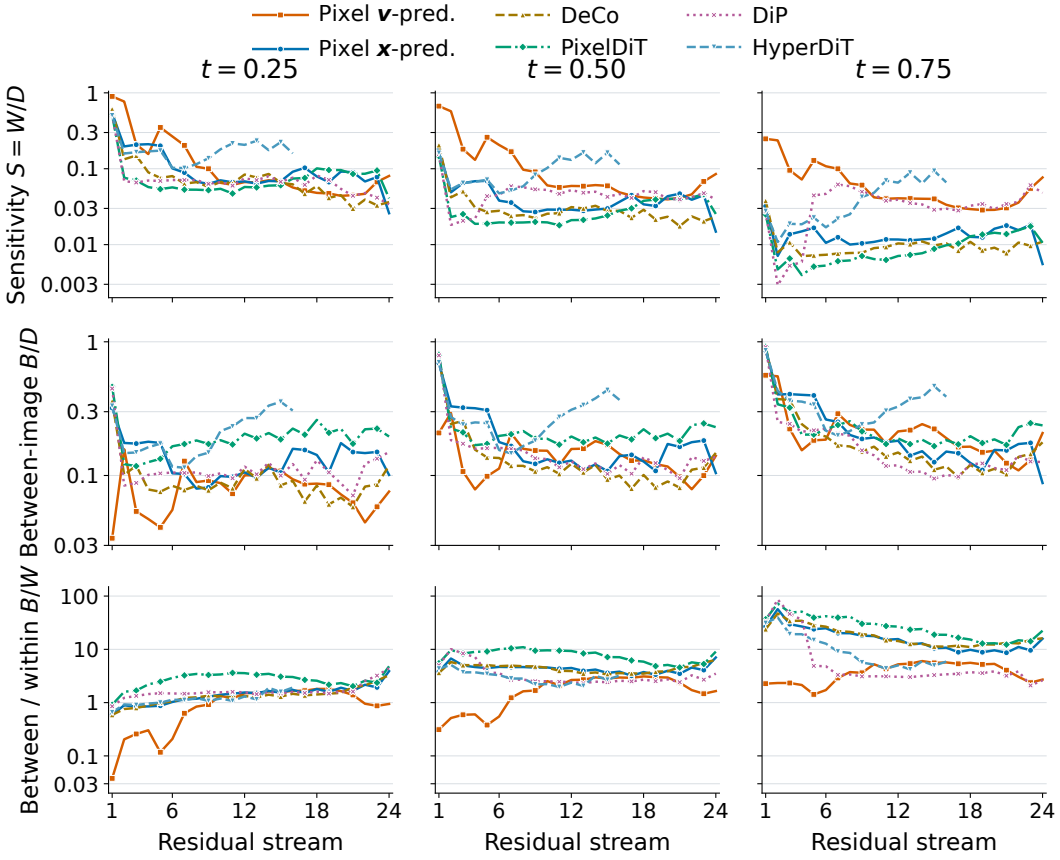


Figure 19: **Image and noise variation evolve differently through depth.** Columns fix the clean coefficient t. The top and middle rows show noise variance W/D and between-image variance B/D, with D the state dimension. The bottom row shows their ratio B/W. Larger ratios indicate stronger image differences relative to corruption. Vertical axes are logarithmic, with a shared range within each row.

Table 15: **Depth-averaged image and noise variation.** Each entry averages a per-observation statistic. In particular, $\bar{R}$ averages the individual ratios rather than dividing the averaged variances. Complete curves appear in Figure 19.

	$t = 0.25$			$t = 0.50$			$t = 0.75$		
Model	$\bar{S}$	$\overline{B/D}$	$\bar{R}$	$\bar{S}$	$\overline{B/D}$	$\bar{R}$	$\bar{S}$	$\overline{B/D}$	$\bar{R}$
Pixel, $\boldsymbol{v}$	0.164	0.081	1.04	0.133	0.142	1.89	0.071	0.221	3.90
Pixel, $\boldsymbol{x}$	0.116	0.137	1.50	0.045	0.192	4.39	0.014	0.234	17.69
DeCo	0.090	0.098	1.41	0.035	0.149	4.44	0.010	0.196	19.07
PixelDiT	0.086	0.197	2.78	0.032	0.224	7.87	0.010	0.241	28.83
DiP	0.083	0.121	1.72	0.049	0.162	3.56	0.036	0.182	11.24
HyperDiT	0.189	0.216	1.19	0.099	0.293	3.07	0.044	0.357	12.25

C.4 SiHC WORKSPACE SENSITIVITY

The 4×4 SiHC model follows Table 14. We observe its read workspace immediately before each attention or MLP update, prior to native normalization and modulation, and use the corresponding inputs for the plain DiT. Both have $D = 256 \times 768$ workspace coordinates.

The next-layer linear probes in Appendix H observe block 2–12 attention inputs, after blocks 1–11. Sensitivity retains all spatial coordinates, whereas the linear probe spatially averages normalized tokens. These measurements share a representation location but test different properties. Figure 20 includes all attention and MLP inputs and separates noise variance from its ratio to image variation.

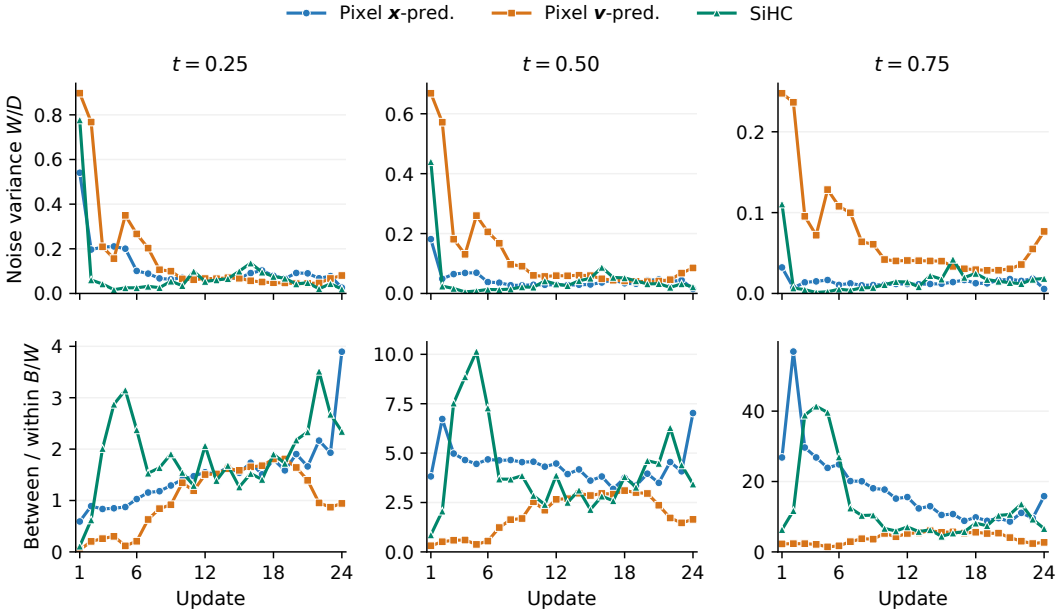


Figure 20: **SiHC's late workspace is less sensitive to corruption than pixel v-prediction.** Columns fix t. Top: noise-induced variance per coordinate (W/D). Bottom: between-image variation relative to noise (B/W). Vertical ranges differ between panels.

At the final MLP input, SiHC has lower noise variance and higher B/W than pixel v-prediction at all three measured times. For example, at $t = 0.50$, their W/D values are 0.0219 and 0.0852, and their B/W values are 3.43 and 1.65, respectively. Both image and noise variation are lower in SiHC at this location, with a larger relative reduction in noise.

D RESIDUAL STATES IN THE SWISS-ROLL TOY

We use the toy setting of Li and He (2026), with a planar Swiss roll embedded in 512 dimensions, and train the same fully connected architecture under x- and v-prediction. Table 16 lists the data, training and sampling settings.

Table 16: **Swiss-roll training and sampling settings.**

Setting	Value
Dataset / ambient dimension	8,192 points / 512; dataset seed 42
Construction	Keep coordinates 0 and 2 of the Swiss roll, project with a Gaussian-QR 512×2 basis, standardize each ambient coordinate
Standardization	Subtract empirical mean, divide by empirical standard deviation plus 10^{-6}
Training	200,000 updates, batch 256, AdamW, constant learning rate 10^{-4}
Optimizer	$\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0, gradient norm clip 1
Time	Sigmoid-normal; implementation noise time $q = 1 - t$, minimum $q = 0.001$
Generation	2,048 standard-Gaussian initial points; raw final weights
Integrator	Euler on 100 evenly spaced time points from $q = 0.999$ to 0.001 (99 updates)
Implementation	JAX

Model. The fully connected denoiser uses five AdaLN-zero residual ReLU blocks with residual state and MLP hidden width 256.

State measurements. The rank measurements use the final 200k checkpoint and all 8,192 saved training examples, with one paired Gaussian draw per example. They record the persistent state before AdaLN at the input of each of the five blocks, so block indices 0–4 in Figure 1 correspond to the inputs of blocks 1–5. Original toy time q corresponds to paper time $t = 1 - q$.

E IMAGENET TRAINING, SAMPLING AND MODEL CONFIGURATIONS

E.1 DATA AND TRAINING

We train on ImageNet-1k (Deng et al., 2009) at 256×256 resolution, with 1,281,167 images and 1,000 classes. We use ADM-style center cropping and random horizontal flips, and scale pixel values to $[-1, 1]$. Table 17 summarizes the shared pixel-space training settings.

Table 17: **Pixel-space training settings.** B/L/H scaling uses 200 epochs without representation alignment. The extended H and XL runs use 600 epochs.

Setting	Value
Image resolution	256×256 RGB
Training duration	200 epochs (controlled/scaling); 600 (extended H/XL)
Optimizer	AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$
Global batch size	1,024
Learning rate	2×10^{-4}, constant after warmup
Weight decay	0
Warmup	5 epochs
Time distribution	$\mathrm{logit}(t) \sim \mathcal{N}(-0.8, 0.8^2)$
Gaussian noise scale	1.0
Division floor	$\max(1 - t, 0.05)$
Class-conditioning dropout	0.1
EMA decay	0.9999
Representation alignment	None for controlled runs and B/L/H scaling; REPA for XL

Prediction and loss. For a clean image $\boldsymbol{x}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$, we form $\boldsymbol{x}_t = t\boldsymbol{x} + (1 - t)\boldsymbol{\epsilon}$. $\boldsymbol{v}$-prediction models minimize mean squared error to $(\boldsymbol{x} - \boldsymbol{x}_t)/\max(1 - t, 0.05)$. $\boldsymbol{x}$-predictions are converted to velocity with the same denominator before computing the loss. The division floor stabilizes training near $t = 1$.

Optimization. Controlled models use the optimizer and learning-rate schedule in Table 17 and train for 200 epochs. Generation uses EMA weights, while class-conditioning dropout trains the unconditional branch used by classifier-free guidance.

Model configurations. Both $\boldsymbol{x}$- and $\boldsymbol{v}$-prediction JiT-B/16 baselines use a direct linear patch embedding without a bottleneck and omit in-context class tokens. Their backbone has 12 blocks, width 768 and 12 attention heads. The main SiHC-B ablations likewise omit in-context tokens. For scaling, all SiHC sizes use 32 in-context class tokens, with the configurations in Table 18.

The scaling configurations embed each 4×4 RGB subpatch directly into the hidden dimension and group sixteen local states within each 16×16 computational patch. Independent static feature-wise read/write maps surround every attention and MLP sublayer, with identity carry between states. The workspace contains 256 spatial tokens and grows to 288 tokens when the class tokens are inserted. Class tokens follow ordinary residual updates and do not enter the spatial read/write maps.

E.2 SAMPLING AND EVALUATION

For pixel sampling, the conditional and null-class predictions define the guided velocity $\widehat{v}_g = \widehat{v}_\varnothing + w(t)(\widehat{v}_y - \widehat{v}_\varnothing)$, where $w(t)$ is the listed CFG scale inside the guidance interval and 1 outside. Standard Gaussian initial states have noise scale one. FID uses 50,000 generated images, balanced

Table 18: **Configurations of SiHC models.** Block indices are one-based. Parameter counts and GFLOPs cover the denoiser, excluding the REPA encoder and projector.

	SiHC-B	SiHC-L	SiHC-XL	SiHC-H
Depth	12	24	40	32
Hidden dimension	768	1,024	1,024	1,280
Attention heads	12	16	16	16
First block with context	5	9	9	9
REPA	No	No	Yes	No
Parameters (M)	131.0	459.5	762.8	953.8
GFLOPs/forward	50.9	177.0	299.1	368.2
Image resolution	256×256			
Computational / local patch	16×16 / 4×4			
Streams per computational patch	16			
In-context class tokens	32			
Input embedding	Direct linear map, no bottleneck			
Read/write maps	Static feature-wise, per sublayer			
Carry	Identity			

Table 19: **Generation protocols for our experiments.** All rows use 50,000 samples. Guidance intervals are in the paper's clean-time convention.

Experiment	Solver	EMA	CFG	Interval
Pixel B-size / ablations	Heun50	0.9999	2.9	$[0.1, 1]$
SiHC-L	Heun50	0.9999	2.4	$[0.1, 1]$
SiHC-H	Heun50	0.9999	2.1	$[0.1, 0.9]$
SiHC-XL + REPA	Heun50	0.9999	2.4	$[0.1, 0.9]$
Whitened $\boldsymbol{x}$ / $\boldsymbol{v}$	Heun50	0.9999	2.9	$[0.1, 1]$
DINO/RAE plain / mHC	Heun50	0.9999	2.9	$[0.1, 1]$

across classes, and the JiT ImageNet-256^2 reference statistics. Samples are converted to uint8 RGB for evaluation. Whitened outputs are inverse-transformed before conversion to RGB. Diagnostic figures use raw weights; generation uses the EMA state specified in Table 19.

B-size decoupled models. Table 20 reports the generation trajectories of our DeCo, DiP and PixelDiT runs used in the decoupled-model analysis. Each uses a 12-block, width-768 semantic backbone, $\boldsymbol{v}$-prediction and 200 training epochs without REPA. We evaluate EMA checkpoints every 40 epochs with the pixel sampling protocol in Table 19. All three improve throughout training, reaching FID 8.07, 13.31 and 6.33, respectively. The corresponding representation diagnostics use raw weights.

Table 20: **Generation trajectories of B-size decoupled models.** FID on 50,000 ImageNet-256^2 samples at the indicated training epochs, using Heun50 and CFG 2.9. All runs use seed 0.

	Training epochs				
Model	40	80	120	160	200
DeCo-B	63.84	19.42	12.64	9.54	8.07
DiP-B	82.13	27.74	19.54	15.89	13.31
PixelDiT-B	67.44	14.83	9.44	7.35	6.33

DINO/RAE training. The shared RAE configuration uses DINOv2-with-registers base, encoder input size 224 and normalized $768 \times 16 \times 16$ latents (Oquab et al., 2024). The frozen ViT-XL decoder and latent normalization follow RAE (Zheng et al., 2026). The decoder noise parameter is zero. Diffusion training uses 200 epochs, global batch 1,024, AdamW with learning rate 2×10^{-4}, betas $(0.9, 0.95)$ and zero weight decay. Learning rate is constant with no warmup, EMA decay is 0.9999 and the training seed is 42. The transport is linear, with logit-normal time sampling and the dimension-dependent time shift used by RAE (latent dimension 196,608, reference dimension 4,096). The mHC variant has four streams and class dropout 0.1. Final evaluation draws 50 samples per class with seed 12345, Heun-50 and CFG 2.9 over $[0.1, 1]$, matching the pixel-space guidance interval. RAE's implementation time runs opposite to the paper's t.

E.3 SiHC ablations

HC intervention. Table 3 uses B-size models trained for 200 epochs. DINOv2-B features have shape $16 \times 16 \times 768$, matching the 256 tokens and 768 coordinates per token obtained by patchifying $256 \times 256 \times 3$ images into 16×16 RGB patches. Each plain/mHC pair shares its training protocol. The mHC intervention uses four copied streams with dynamic read, write and mixing maps, while keeping attention and MLP widths fixed. Both pixel and DINOv2-B models are evaluated with Heun-50 and CFG 2.9 over $[0.1, 1]$.

Design ablations. Table 21 specifies the interfaces behind Table 4. The main SiHC progression uses separate read/write maps for attention and MLP, giving 24 connection events in the 12-block B backbone. All models use computational patch size 16×16, 200 training epochs and the B-size sampling protocol. Adding 32 in-context class tokens, as in JiT, to the 4×4 model lowers FID from 8.07 to 6.17. The lower group separately compares direct and factorized embeddings in the earlier blockwise model, with intermediate width 128 for the latter.

Table 21: **B-size ablation configurations.** All models use $\boldsymbol{v}$-prediction. b is local input side length in pixels, and S is the number of streams per computational patch.

Configuration	b	S	Read/write	Carry	FID
mHC copied input	16	4	mHC	mixed	25.36
mHC spatial input	8	4	mHC	mixed	13.25
Static scalar access	8	4	scalar	identity	11.50
SiHC, spatial input	8	4	feature-wise	identity	10.10
SiHC, direct input	4	16	feature-wise	identity	8.07
+ 32 in-context class tokens	4	16	feature-wise	identity	6.17
Earlier blockwise embedding comparison					
SiHC, direct input	4	16	feature-wise	identity	8.54
Factorized embedding	4	16	feature-wise	identity	7.97

E.4 Scaling without representation alignment

SiHC scales effectively under direct $\boldsymbol{v}$-prediction without a pretrained representation encoder. With 32 in-context class tokens and sublayerwise read/write maps, increasing the backbone from B to L and H reduces FID from 6.17 to 3.02 and 2.51 after 200 epochs (Table 22). Figure 21 shows this progression and the subsequent training of SiHC-H, which levels off around FID 2.06.

Table 22: **Scaling pixel-space Transformers on ImageNet-256^2.** The upper group reproduces DiP's DiT-only results with its Euler-100 protocol (Chen et al., 2026b). The lower group uses 200 training epochs without REPA and Heun-50, with CFG $2.9/2.4/2.1$ for B/L/H. GFLOPs count one denoiser forward with two FLOPs per multiply-add. *Analytical estimates from the published configurations and released backbone (Appendix E.6).

Model	Depth × width	Params	GFLOPs	FID ↓	IS ↑
DiT-only configurations reported by DiP					
DiT-only	26×1152	629M	221.2^*	5.28	243.8
DiT-only	32×1152	772M	272.0^*	4.91	251.7
DiT-only	26×1280	776M	272.0^*	4.28	249.6
DiT-only	26×1536	1.1B	389.3^*	2.83	285.6
SiHC, 200 epochs, without REPA					
SiHC-B	12×768	131M	50.9	6.17	196.3
SiHC-L	24×1024	460M	177.0	3.02	268.8
SiHC-H	32×1280	954M	368.2	2.51	274.1

DiP's DiT-only study reports FID 2.83 for its widest 1.1B-parameter model, alongside our SiHC-H result of 2.51 with 954M parameters under the protocols in Table 22.

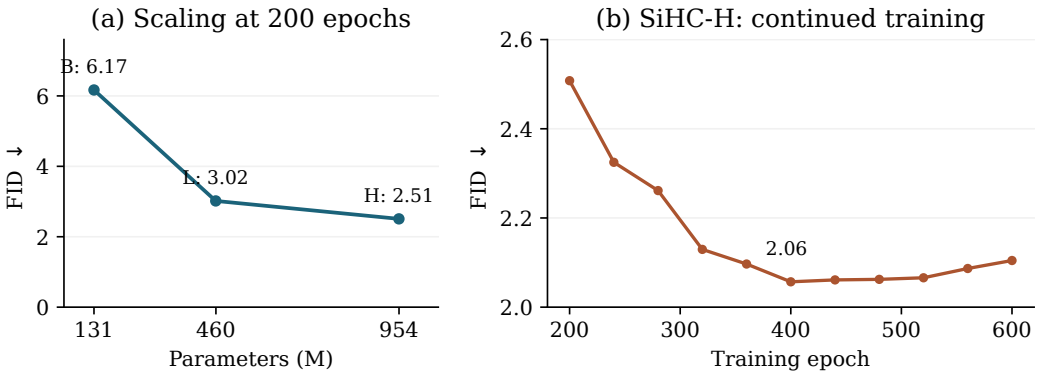


Figure 21: **Scaling SiHC without representation alignment.** Left: B/L/H at 200 epochs, with 32 in-context class tokens and sublayerwise read/write. Right: continued training of SiHC-H, including all measured points from epoch 200 onward. Generation quality levels off around FID 2.06.

E.5 SiHC-XL with representation alignment

Our XL configuration combines the same sublayerwise read/write interface with REPA (Yu et al., 2025) and reaches FID 1.71. It uses 40 blocks of width 1024, direct 4×4 patch embeddings and 32 in-context class tokens inserted before block 9. Training follows Table 17, with a frozen DINOv3 ViT-L/16 teacher (Siméoni et al., 2026). The teacher receives clean 256×256 images with ImageNet normalization. Removing its CLS and four register tokens leaves 256 patch features of dimension 1024.

At block 8, before context tokens enter, we align the sum of the student's MLP read workspace and its gated update. A three-layer projector with dimensions $1024 \to 2048 \to 2048 \to 1024$ and SiLU activations maps these workspace features to the teacher space. The training loss adds 0.5 times the negative cosine similarity, averaged over patches, to the velocity MSE. The teacher and projector are used only during training. The denoiser contains 762,840,400 parameters and uses 299.10 GFLOPs per forward pass. BF16 autocast and Flash attention are used for training, with the static read/write maps fused as described in Appendix G.

Reference results and cost accounting. Table 7 reports guided FID on 50,000 samples, with each method's own recipe and training duration. Parameter counts cover the denoiser, excluding pretrained autoencoders and alignment encoders. RAE uses AutoGuidance for its DiT$^{\mathrm{DH}}$ result.

Table 18 reports parameter counts and forward costs for the listed configurations. FLOPs count dominant matrix operations and leading residual-interface arithmetic, with one multiply-add counted as two FLOPs.

E.6 Compute accounting

Reference-model compute accounting. Table 7 uses the two-FLOPs-per-multiply-add convention for linear, convolution and attention matrix products. DiT, SiT, REPA, LightningDiT and DDT use the published 119-G multiply-add reference, converted to 238 GFLOPs. PixelFlow and PixNerd use 5818 and 268 GFLOPs under the same convention, as reported in Li and He (2026); Chen et al. (2026a). JiT-H/G, DiP-XL, DeCo-XL, PixelDiT-XL and SiHC-XL use the cached architecture counts, rounded to whole GFLOPs. PixelREPA retains the JiT-H inference architecture because its masked adapter is used only during training (Shin et al., 2026).

The RAE DiT-XL and DiT$^{\mathrm{DH}}$-XL counts are 238.12 and 323.16 GFLOPs, computed from the official 256-token architectures, including spatially conditioned modulation in the wide head. HyperDiT-XL's 551.4 GFLOPs are an estimate from the local implementation of its published configuration, including both branches, connectors and registers. RAEv2 uses 875.22M parameters and 475.00 GFLOPs, counted from its released DINOv3-L $K = 7$ configuration, including the internal guidance head and excluding the representation encoder and decoder. AsymFlow uses its reported 363 GFLOPs.

Parameter counts in the table otherwise follow the cited papers. These counts exclude pointwise nonlinearities and normalization, and describe model computation rather than hardware runtime.

Scaling-model costs. For SiHC, the counts include 32 context tokens from the block specified in Table 18 and independent read/write maps at both sublayers. The DiT-only GFLOPs in Table 22 are analytical estimates using DiP's published depths and widths, 256 tokens, the released SwiGLU backbone, and direct 16×16 input/output projections. We count attention products, feed-forward projections and AdaLN modulation under the same convention as SiHC. DiP's parameter counts, FID and IS are taken directly from its Table 2.

Controlled-model costs. For the main progression in Table 4, we count model parameters, including frozen positional parameters where present, and add leading residual-interface arithmetic to the matrix-operation costs. The plain DiT uses 131.69M parameters and 46.60 GFLOPs. Copied-input and spatial mHC use 133.46M / 133.16M parameters, both at 47.73 GFLOPs. SiHC uses 24 sublayer read/write pairs: 8×8 / 4×4 patches give 130,754,496 / 130,975,536 parameters and 46.671 / 46.897 GFLOPs. Adding 32 in-context class tokens from the fifth block gives 131,000,112 parameters and 50.949 GFLOPs: their positional embeddings add 24,576 parameters, and attention/MLP computation on the longer sequence adds 4.052 GFLOPs. The static scalar-map estimate holds the 8×8 SiHC backbone fixed and replaces each pair of $C \times S$ feature-wise maps with two S-entry scalar maps. This gives 130,607,232 parameters and the same 46.671 GFLOPs. Feature-wise access adds 147,264 parameters while operating on the same residual-state coordinates.

E.7 Linear-interface arithmetic cost

A feature-wise read/write pair uses $2CS$ parameters and $4MCS$ leading FLOPs, counting a multiply-add as two FLOPs. A depth-L sublayerwise model has $2L$ pairs, totaling $4LCS$ parameters and $8LMCS$ FLOPs. With $L = 12$, $C = 768$, $S = 16$ and $M = 256$, these are 0.590M parameters and 0.302G FLOPs. Scalar maps share coefficients across channels, reducing the interface parameter count to $4LS$ while retaining the same leading arithmetic over MSC activations. The earlier blockwise variant has half as many read/write pairs (Appendix F.1). These counts use the unfused recurrence to compare architectures under a common convention; stage fusion can change the executed arithmetic (Appendix G). For mHC, matrix-operation counting already includes the read, so the interface correction adds only the previously uncounted write contribution. Normalization and pointwise nonlinearities are excluded throughout.

F Read/Write Interface and Residual Paths

SiHC reads from and writes to its spatial states around each attention and MLP sublayer. We also specify the earlier blockwise form used for the embedding comparison in Table 21 (Figure 22).

Feature-wise read and write. With elementwise multiplication $\odot$, Equation (3) becomes

$$\boldsymbol{h}_\ell^{\mathrm{in}} = \sum_s \boldsymbol{\mathcal{H}}_{\mathrm{pre}}^\ell[:, s] \odot \boldsymbol{h}_\ell^{(s)}, \tag{21}$$

$$\boldsymbol{h}_{\ell+1}^{(s)} = \boldsymbol{h}_\ell^{(s)} + \boldsymbol{\mathcal{H}}_{\mathrm{post}}^\ell[:, s] \odot \Delta \boldsymbol{h}_\ell. \tag{22}$$

Each $C \times S$ map assigns a separate coefficient to every feature and stream.

F.1 Sublayerwise updates and the earlier blockwise form

Let $\mathcal{R}_\ell$ and $\mathcal{P}_\ell$ denote the feature-wise read and write above. Write $\mathcal{A}_\ell$ and $\mathcal{M}_\ell$ for the gated attention and MLP updates, including their normalization and time/class conditioning.

Sublayerwise. Attention and MLP each have their own read/write pair:

$$\boldsymbol{z}_\ell^{\mathrm{A}} = \mathcal{R}_\ell^{\mathrm{A}}(\boldsymbol{H}_\ell), \qquad \boldsymbol{H}_{\ell+\frac{1}{2}} = \boldsymbol{H}_\ell + \mathcal{P}_\ell^{\mathrm{A}}\big(\mathcal{A}_\ell(\boldsymbol{z}_\ell^{\mathrm{A}})\big), \tag{23}$$

$$\boldsymbol{z}_\ell^{\mathrm{M}} = \mathcal{R}_\ell^{\mathrm{M}}(\boldsymbol{H}_{\ell+\frac{1}{2}}), \qquad \boldsymbol{H}_{\ell+1} = \boldsymbol{H}_{\ell+\frac{1}{2}} + \mathcal{P}_\ell^{\mathrm{M}}\big(\mathcal{M}_\ell(\boldsymbol{z}_\ell^{\mathrm{M}})\big). \tag{24}$$

The MLP reads the attention-updated states through its own map, giving $2L$ connection events for L blocks.

Earlier blockwise form. One read/write pair surrounds the combined attention and MLP update:

$$\boldsymbol{z}_\ell = \mathcal{R}_\ell(\boldsymbol{H}_\ell), \qquad \boldsymbol{a}_\ell = \mathcal{A}_\ell(\boldsymbol{z}_\ell), \tag{25}$$

$$\boldsymbol{m}_\ell = \mathcal{M}_\ell(\boldsymbol{z}_\ell + \boldsymbol{a}_\ell), \qquad \boldsymbol{H}_{\ell+1} = \boldsymbol{H}_\ell + \mathcal{P}_\ell(\boldsymbol{a}_\ell + \boldsymbol{m}_\ell). \tag{26}$$

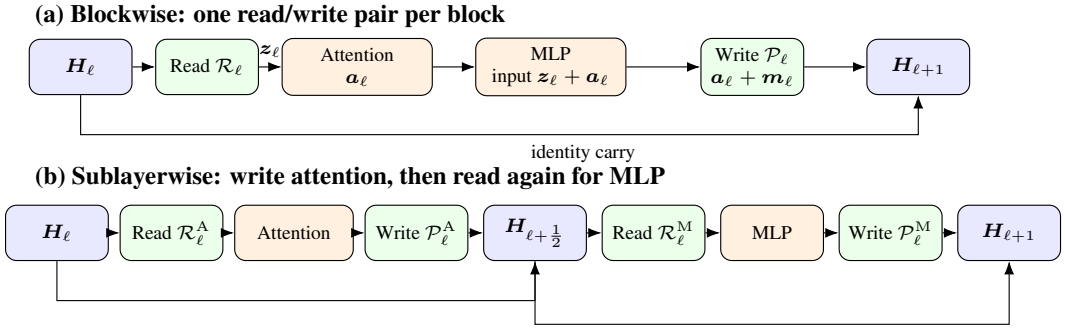


Figure 22: **Two execution granularities of SiHC.** Blue boxes are persistent spatial states; orange boxes operate on the compact workspace. Blockwise SiHC writes the combined attention/MLP update once. Sublayerwise SiHC writes after attention and reads again for the MLP. Both use identity carry and the same spatial input/output assignment.

F.2 Gradients through one connection event

The following derivatives apply to each connection event, with ℓ indexing events and $m = 1, \ldots, M$ indexing computational cells. For blockwise SiHC, $\mathcal{F}_\ell$ combines attention and MLP updates, whereas for sublayerwise SiHC it denotes one of these updates.

Forward and backward paths. Let $g_\ell^{(s)}(m, c) = \partial \mathcal{L} / \partial h_{\ell,c}^{(s)}(m)$ be a scalar loss derivative. The gradient entering the nonlinear update and the gradient returned to its input satisfy

$$q_\ell(m, c) = \sum_s \boldsymbol{\mathcal{H}}_{\mathrm{post}}^\ell[c, s] g_{\ell+1}^{(s)}(m, c), \qquad \boldsymbol{r}_\ell = \boldsymbol{J}_\ell^\top \boldsymbol{q}_\ell. \tag{27}$$

The vectors $\boldsymbol{q}_\ell, \boldsymbol{r}_\ell \in \mathbb{R}^{MC}$ stack all cells and features, and $\boldsymbol{J}_\ell \in \mathbb{R}^{MC \times MC}$ is the update Jacobian from its stacked read vectors to its stacked updates. This includes attention's interactions across cells. Identity carry then gives

$$g_\ell^{(s)}(m, c) = g_{\ell+1}^{(s)}(m, c) + \boldsymbol{\mathcal{H}}_{\mathrm{pre}}^\ell[c, s] r_\ell(m, c). \tag{28}$$

A plain residual connection already has an identity path. SiHC widens the state preserved by this path while passing its nonlinear interaction through the smaller workspace.

G Kernel Fusion by Exact Stage-Wise Reordering

G.1 From repeated wide-state updates to a compact recurrence

A literal SiHC update reads an S-stream state, computes a width-C update, and writes the result back to every stream. The Transformer uses a compact workspace, but materializing each intermediate state repeatedly moves the wider SC representation. Static feature-wise maps and identity carry let us collect these writes over a stage and evaluate the same recurrence through the compact updates.

For either granularity in Appendix F.1, let a stage contain T residual updates, indexed by $\ell = 0, \ldots, T-1$. Flatten batch and computational-cell indices into $q = 1, \ldots, Q$, where $Q = B_{\mathrm{batch}} M$.

Write $X_{\ell,qsc}$ for the retained state, $A_{\ell cs} = \boldsymbol{\mathcal{H}}^{\ell}_{\mathrm{pre}}[c,s]$ and $P_{\ell cs} = \boldsymbol{\mathcal{H}}^{\ell}_{\mathrm{post}}[c,s]$. The literal operations are

$$Z_{\ell qc} = \sum_s A_{\ell cs} X_{\ell qsc}, \qquad \Delta Z_\ell = \mathcal{F}_\ell(Z_\ell, t), \qquad X_{\ell+1,qsc} = X_{\ell qsc} + P_{\ell cs} \Delta Z_{\ell qc}. \tag{29}$$

Unrolling the identity carry gives $X_{\ell qsc} = X_{0qsc} + \sum_{j<\ell} P_{jcs} \Delta Z_{jqc}$. Substitute this into the read and define

$$B_{\ell qc} = \sum_s A_{\ell cs} X_{0qsc}, \qquad \Gamma_{\ell jc} = \sum_s A_{\ell cs} P_{jcs} \quad (j < \ell). \tag{30}$$

Then the forward pass is

$$Z_{\ell qc} = B_{\ell qc} + \sum_{j<\ell} \Gamma_{\ell jc} \Delta Z_{jqc}, \qquad \Delta Z_\ell = \mathcal{F}_\ell(Z_\ell, t), \tag{31}$$

$$X_{\mathrm{out},qsc} = X_{0qsc} + \sum_{\ell=0}^{T-1} P_{\ell cs} \Delta Z_{\ell qc}. \tag{32}$$

The first equation computes each compact workspace in sequence, and the second reconstructs all spatial states at the stage boundary (Figure 23). Since each workspace is the same algebraic function of preceding updates, attention and MLP computation receive the same inputs as in literal execution. Floating-point summation follows the fused reduction order.

This is *late materialization*: each spatial state remains logically defined by its stage input and accumulated writes, while subsequent reads are evaluated directly from that representation. The expanded residual-stream bandwidth is therefore available without storing a separate wide tensor at every update. At the stage boundary, the final kernel accumulates each output tile on chip and writes it back once, avoiding the repeated wide-state transfers of literal execution.

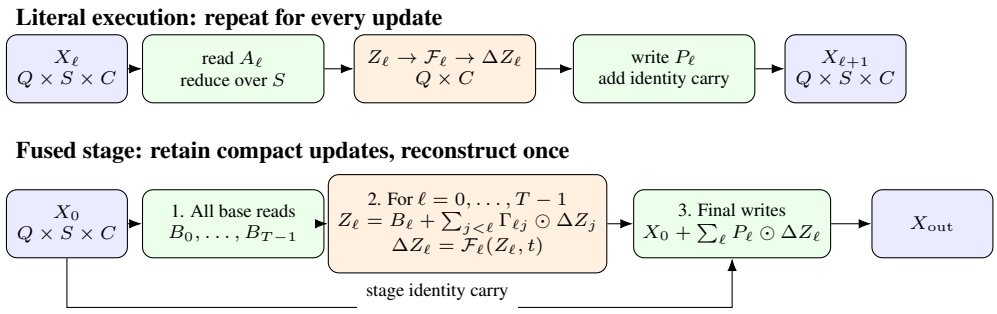


Figure 23: **Fusion removes intermediate wide-state materialization.** Blue tensors hold the S persistent streams; orange tensors have workspace width C. All base reads are computed from the same stage input. The triangular recurrence restores the effect of earlier writes before each nonlinear update. Only the final operation reconstructs the wide state.

G.2 Three operation families and stage boundaries

Table 23 gives the tensors and launch responsibilities. With 4×4 subpatches, the ImageNet state layout is $B_{\mathrm{batch}} \times 64 \times 64 \times C$, regrouped into $M = 256$ cells with $S = 16$ states each. Read and write coefficients are shared across cells and samples. They initialize to $A_{\ell cs} = 1/S$ and $P_{\ell cs} = 1$ and are learned independently at every update.

The CUDA fast path implements the base reads, active triangular sums and final accumulation as custom Triton operations. A distinct active-prefix operation fixes the number of preceding updates at compile time. The final kernel accumulates all writes in registers before storing each output element. Attention, RMSNorm, AdaLN and SwiGLU remain standard PyTorch operations captured by `torch.compile`. Thus one stage contains multiple launches and sequential nonlinear updates.

The blockwise kernels specialize to 8- and 12-block stages. The sublayerwise variant has independent attention and MLP connections and uses 16- and 24-event kernels for the same stage lengths. Its T therefore counts twice as many connection events. At a stage boundary, X_{out} becomes the next X_0, resetting the update history. In-context class tokens are introduced at their configured start block. Stage sizes, connection granularity and context insertion are part of the model configuration.

Table 23: **Fused interface operations.** T counts residual updates in one stage and $Q = B_{\text{batch}}M$.

Operation	Computation / implementation	Output shape
Coefficient contraction	Reduce $A_\ell P_j$ over streams for $j < \ell$	$T \times T \times C$
All base reads	Read X_0 with every A_ℓ; separate outputs feed compiled consumers	T tensors $Q \times C$
Active triangular sum	At update ℓ, accumulate the previous ℓ updates with $\Gamma_{\ell j}$	$Q \times C$
Nonlinear update	Attention/MLP, normalization and conditioning on the compact workspace	$Q \times C$
Final accumulation	Reduce all writes, add X_0, store the final spatial states	$Q \times S \times C$

Numerical accumulation. Forward reductions accumulate in FP32 and store results in the workspace dtype. Read/write coefficient gradients also accumulate in FP32 across samples and spatial positions.

G.3 Backward propagation through the fused recurrence

Gradients pass through the base reads, the triangular recurrence and the final write. For a loss applied to the stage output, let $\bar{X}_{qsc} = \partial\mathcal{L}/\partial X_{\text{out},qsc}$ and let $G_{\ell qc} = \partial\mathcal{L}/\partial Z_{\ell qc}$ denote the total read gradient after reversing all later updates. Reverse execution visits $\ell = T-1, \dots, 0$. The gradient of the compact update is

$$D_{\ell qc} = \sum_s P_{\ell cs}\bar{X}_{qsc} + \sum_{k>\ell} \Gamma_{k\ell c} G_{kqc}, \qquad G_\ell = \boldsymbol{J}_\ell^\top D_\ell, \tag{33}$$

where $\boldsymbol{J}_\ell$ is the full nonlinear-update Jacobian, including attention across cells. The interface gradients are

$$\bar{\Gamma}_{\ell jc} = \sum_q G_{\ell qc}\Delta Z_{jqc}, \quad j < \ell, \tag{34}$$

$$\frac{\partial\mathcal{L}}{\partial X_{0qsc}} = \bar{X}_{qsc} + \sum_\ell A_{\ell cs} G_{\ell qc}, \tag{35}$$

$$\frac{\partial\mathcal{L}}{\partial A_{\ell cs}} = \sum_q G_{\ell qc} X_{0qsc} + \sum_{j<\ell} \bar{\Gamma}_{\ell jc} P_{jcs}, \tag{36}$$

$$\frac{\partial\mathcal{L}}{\partial P_{jcs}} = \sum_q \bar{X}_{qsc}\Delta Z_{jqc} + \sum_{\ell>j} \bar{\Gamma}_{\ell jc} A_{\ell cs}. \tag{37}$$

These expressions identify the reductions required by the custom backward kernels. In particular, Γ remains connected to both learned maps, so its gradients train the same read/write coefficients as literal execution. Checkpointing nonlinear activations can be combined with this interface schedule.

G.4 Storage and arithmetic

Literal autograd may retain T wide states for its read-weight gradients, costing $TQSC$ elements. The reordered interface stores the stage input and width-C base reads/updates, giving $O(QSC + TQC + T^2C)$ interface storage. Nonlinear attention/MLP activations are additional to both.

Fusion reduces intermediate state materialization and reorganizes memory access and launches. The leading interface work, counting a multiply-add as two FLOPs, is approximately

$$4TQCS + T(T-1)QC + T(T-1)CS. \tag{38}$$

The first term computes base reads and final writes. The second is the triangular accumulation, and the third constructs its coefficients. Literal read/write execution has leading cost $4TQCS$. Short stages bound the additional triangular work. In the fused final accumulation, the wide input is read and the output is stored once; base-read kernels also access the stage input.

G.5 Measured runtime and memory

Table 24 compares literal PyTorch routing with fused routing under `torch.compile`, holding topology, size and recomputation fixed. Models use 256×256 inputs, 4×4 subpatches, a 16×16 workspace

grid and 32 in-context tokens. B/L/XL/H have 12/24/40/32 blocks of width 768/1024/1024/1280. Both implementations use no recomputation for B/L and MLP recomputation for XL/H.

Measurements use one H100 80GB per trial, batch size 128, FP32 parameters, BF16 autocast and Flash SDPA, with PyTorch 2.9.1+cu128 and Triton 3.5.1. Training measures forward, loss, backward, fused AdamW and two FP32 EMA updates under default compilation and one-rank DDP. Inference measures one denoiser forward under `reduce-overhead` compilation. Each entry is the median of three fresh-process trials with eight warmup and 50 measured steps, excluding compilation, data loading, checkpoint I/O and REPA. Memory is peak allocated memory after training warmup, or including CUDA Graph setup for inference.

Table 24: **Compiled SiHC with and without fused routing.** Batch size 128 on H100 80GB. Training measures an optimizer step with two EMAs, and inference measures one denoiser forward. Memory is peak allocated GiB. OOM denotes failure at the same requested batch size.

		Training		Inference	
Size	Routing	ms	GiB	ms	GiB
Blockwise					
B	Torch	123.42	26.82	35.90	3.05
B	Fused	97.05	18.90	26.54	3.01
L	Torch	365.24	70.83	102.17	5.05
L	Fused	292.77	50.33	76.76	4.99
XL	Torch	OOM	OOM	171.47	6.19
XL	Fused	527.67	59.04	129.73	6.13
H	Torch	OOM	OOM	191.16	7.71
H	Fused	602.97	63.06	149.61	7.60
Sublayerwise					
B	Torch	210.38	30.18	52.27	3.68
B	Fused	113.73	19.82	28.82	3.25
L	Torch	OOM	OOM	146.50	5.93
L	Fused	330.64	52.07	83.48	5.44
XL	Torch	OOM	OOM	247.14	7.07
XL	Fused	573.15	62.07	137.36	6.08
H	Torch	OOM	OOM	266.41	8.86
H	Fused	682.42	67.18	157.72	8.19

For sublayerwise B, fusion gives a $1.85\times$ training-step speedup and reduces peak allocated memory from 30.18 to 19.82 GiB. Single-forward inference speedups range from $1.69\times$ to $1.81\times$ across the four sublayerwise sizes. The anonymous source package provides reference implementations and numerical tests for the fused forward and backward operations.

H Linear Probes of Next-Layer Inputs

The probes measure how readily an ImageNet class can be decoded from the representation supplied to the next attention sublayer. They use frozen epoch-200 raw backbone weights, with true labels supplied only to the linear classifiers. All backbones receive null class conditioning. There is no additional noise injected into hidden states.

Input and time conventions. We form $\boldsymbol{z} = (1-\alpha)\boldsymbol{x} + \alpha\boldsymbol{\epsilon}$ at noise coefficients $\alpha \in \{0.25, 0.5, 0.75\}$. Pixel inputs are RGB images in $[-1, 1]$, and RAE inputs are normalized DINOv2-B latents. The paper's clean coefficient is $t = 1 - \alpha$. The pixel implementations receive this t, while the RAE implementation uses the opposite time convention and receives α. Each noise level has independently trained probes. The main results evaluate the same noise level used during probe training.

Extraction and fitting. At each boundary after blocks 1–11, the plain DiT supplies its residual state, while SiHC supplies the next attention read workspace $\{\boldsymbol{h}_\ell^{\mathrm{in}}(m)\}_{m=1}^{M}$. All are recorded before AdaLN. We apply non-affine RMS normalization to each patch token, average the tokens spatially, and fit an independent affine $768 \to 1000$ classifier at every boundary. Boundary 11 precedes block

12 and is the fixed boundary used in Table 26. All its entries use matched input noise, and SiHC assigns states to 4×4 subpatches.

Table 25: **Linear-probe fitting and evaluation.** Every boundary and noise level has an independent affine classifier.

Setting	Hidden-state probes
Frozen backbone	Raw epoch 200; null-class conditioning
Data	All 1,281,167 training and 50,000 validation images
Image preprocessing	ADM center crop; no random horizontal flips
Noise coefficient	$\alpha = 0.25, 0.50, 0.75$, with paper time $t = 1 - \alpha$
Feature / head	Per-token RMS (10^{-6}) then GAP; affine $768 \to 1000$
Training	40 epochs; batch size 4,096
Optimizer	AdamW, default betas $(0.9, 0.999)$, zero weight decay
Learning rate	Cosine from 0.01 to 0.0001
Training noise	Fresh Gaussian noise each epoch; paired shuffle and noise across models
Random seeds	Training base 700001; validation 800001, 800002, 800003
Precision	BF16 backbone; FP32 normalization, pooling and classifiers
Evaluation	Matched training/validation noise coefficient; mean and SD across three noise draws

Table 26: **State organization and prediction space shape class readability.** Entries give Top-1 / Top-5 accuracy (%) at the last measured boundary. Columns increase input noise, and larger values indicate more linearly readable class information.

Model	25% noise	50% noise	75% noise
Pixel, $\boldsymbol{x}$	25.32 / 46.45	27.54 / 49.64	22.82 / 43.91
Pixel, $\boldsymbol{v}$	5.38 / 14.51	4.56 / 12.68	3.47 / 10.17
Pixel SiHC, $\boldsymbol{v}$	25.51 / 46.68	27.81 / 50.32	22.88 / 43.98
DINO, $\boldsymbol{x}$	80.15 / 95.47	80.09 / 95.41	80.06 / 95.36
DINO, $\boldsymbol{v}$	81.09 / 95.74	81.07 / 95.81	80.93 / 95.77

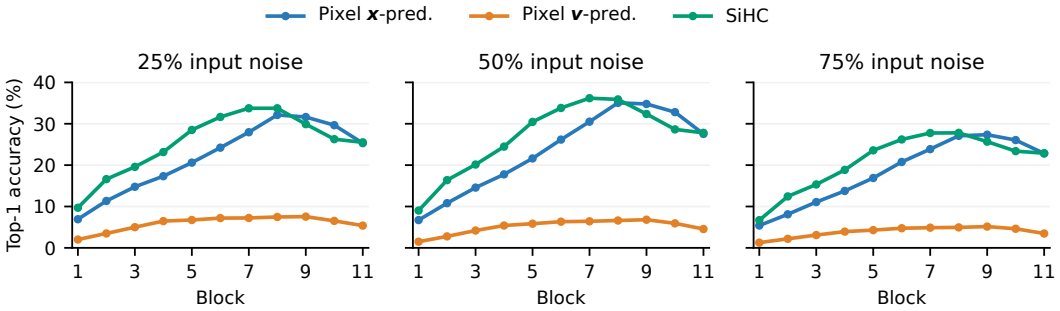


Figure 24: **SiHC recovers clean-like class readability while predicting velocity.** Panels fix the input-noise level. Linear-probe Top-1 accuracy measures class information through depth: SiHC approaches pixel $\boldsymbol{x}$-prediction and remains well above pixel $\boldsymbol{v}$-prediction. Shading denotes standard deviation across noise draws.

At 50% input noise, the depth maxima of SiHC, pixel $\boldsymbol{x}$-prediction and pixel $\boldsymbol{v}$-prediction in Figure 24 are 36.18%, 35.05%, and 6.80%, respectively.

Whitened and decoupled models. Whitened models are probed in their own training space with the same protocol in Table 25. Decoupled models expose the semantic state before the next block. The plain DiT, SiHC, DeCo, PixelDiT and DiP have 11 measured boundaries; the eight-block HyperDiT has seven. All features are recorded before the next attention normalization and modulation. Figure 25 pairs the decoupled probes with their B/W ratios.

Class readability in DINO space. Both plain DINO models have final-boundary probe accuracies above 80% at all three measured noise levels (Table 26). Figure 26 compares their depth profiles with pixel-space models, using the same extraction and fitting protocol.

At 50% input noise, the final measured probe of DINO $\boldsymbol{v}$-prediction reaches 81.07% Top-1 accuracy, compared with 4.56% for pixel $\boldsymbol{v}$-prediction. The DINO input control reaches 80.49% before the denoiser (Tables 26 and 27).

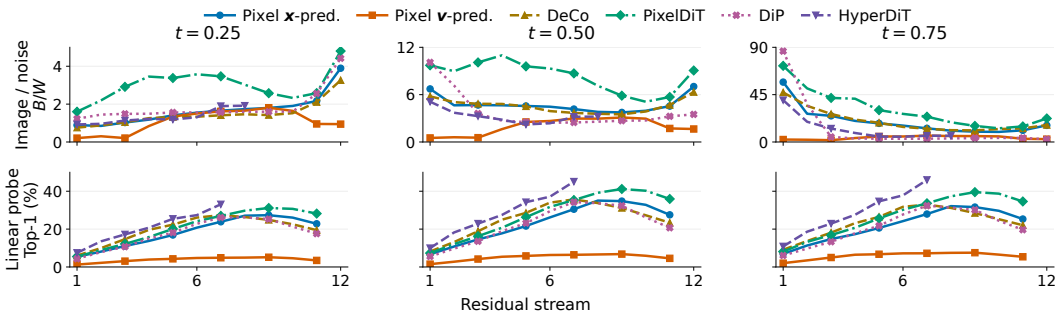


Figure 25: **Separate pixel paths support stronger semantic representations.** Columns fix the clean coefficient t. Top: between-image variation relative to noise (B/W). Bottom: linear-probe Top-1 accuracy through depth. Compare the decoupled $\boldsymbol{v}$-prediction models with pixel $\boldsymbol{x}$- and $\boldsymbol{v}$-prediction. Shading denotes standard deviation across noise draws.

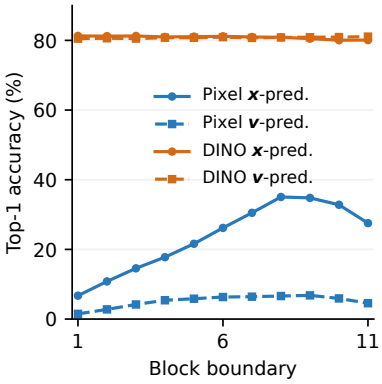


Figure 26: **DINO-space layer inputs retain high class readability under both prediction targets.** Linear-probe accuracy follows block boundaries for DINO (orange) and pixel (blue) features. Solid lines use $\boldsymbol{x}$-prediction and dashed lines use $\boldsymbol{v}$-prediction. All curves use 50% input noise. Shading denotes standard deviation across noise draws. The input control is reported in Table 27.

DINO input control. The input control uses the official RAE encoder with DINOv2-with-registers base. Encoding resizes to 224, discards the class and register tokens, applies the encoder's non-affine final LayerNorm and the RAE latent normalization, and returns 256×768 features. The classifier acts on their spatial mean. Corruption is applied to the encoded latent features.

Table 27: **Noisy DINO inputs retain strong class information.** Rows increase input noise. Top-1 and Top-5 measure linear-probe accuracy, and $\pm$ denotes standard deviation across noise draws.

Matched input noise	Top-1 (%)	Top-5 (%)
0%	79.372	95.036
25%	$79.872 \pm .037$	$95.267 \pm .011$
50%	$80.485 \pm .072$	$95.545 \pm .011$
75%	$80.820 \pm .088$	$95.847 \pm .042$

The input-control probes use seed 42 and four zero-initialized affine classifiers, one for each noise level including zero noise. They train for 40 epochs with batch 4,096, AdamW betas $(0.9, 0.999)$, $\epsilon = 10^{-8}$, zero weight decay and cosine learning rate from 0.01 to 0.0001. Computation is FP32 with TF32 disabled. Evaluation uses the fixed epoch-40 checkpoint and three matched-noise draws on all 50,000 validation images. The input readout is direct GAP of normalized RAE latents.