DEPTH AS TIME IN ONE-STEP GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The recent wave of one-step generative models, which compress the multi-step trajectory of diffusion via either distillation or learned flow maps, has reached an inflection point where they can generate high-quality images. Here, we ask a natural question that follows from these advances: what happens to the denoising trajectory of multi-step diffusion when generation is compressed into a single forward pass? We offer an empirical observation we call *depth as time*: the denoising computation that multi-step diffusion performs across sampling steps appears to unfold across the depth of a single forward pass, and can be recovered by decoding intermediate layers with the model's own output head. Most interestingly, we show that this depthwise computation depends on the transport task a flow map is trained to solve. The most surprising case is MeanFlow, where probing shorter transport intervals reveals both denoising and renoising within a single network evaluation. In contrast, generators trained without a time-indexed transport task, such as drifting models, do not exhibit the same depthwise denoising. Consequently, we show that models that exhibit the depthwise denoising phenomenon are more compressible across the layerwise computation: a MeanFlow `SiT-L/2` model can be compressed by $16.6\times$ in parameters into a single time-conditioned block. Together, these results suggest that the temporal computation of diffusion is not eliminated by one-step generation, but reorganized across network depth. Code will be available upon acceptance.

1 INTRODUCTION

One-step generative models based on diffusion have recently been shown to produce high-quality images without the iterative denoising trajectory of multi-step diffusion, using methods ranging from distillation (Salimans and Ho, 2022; Luo et al., 2023; Yin et al., 2024; Zhou et al., 2024) to directly learning flow maps (Song et al., 2023; Frans et al., 2025; Boffi et al., 2025; Geng et al., 2025). This efficiency, however, seems to remove a defining feature of diffusion- and flow-based generation: the explicit trajectory of intermediate states that evolves from coarse to fine-grained detail. This trajectory provides a natural axis for intervention and control via guidance methods that modify the generative vector field depending on the desired goal. Several downstream advantages of multi-step diffusion have followed, ranging from classifier-free guidance (Ho and Salimans, 2022; Wang et al., 2026), reinforcement-learning-based control (Black et al., 2024; Wallace et al., 2024; Fan et al., 2023), and image editing (Meng et al., 2021; Lee et al., 2026; Liang et al., 2026) to classification (Li et al., 2023; Clark and Jaini, 2023), fine-grained generation (Guo et al., 2026; Zarei et al., 2026), and many others.

It is natural, therefore, to ask what happens to this denoising trajectory when generation is compressed into a single forward pass. We show that distilled one-step models and flow maps effectively *unroll* this loop across network depth, and that decoding their intermediate blocks reveals a coarse-to-fine progression reminiscent of the one multi-step diffusion exposes across sampling steps. We call this observation *depth as time*. In contrast, one-step models not based on diffusion, such as drifting models (Deng et al., 2026), do not appear to exhibit the same transport-organized depthwise trajectory.

Surprisingly, depth as time can be observed without any additional training. Applying the model's own output head to its intermediate representations recovers a layerwise velocity (Sec. 3.1), from which we decode the image at each depth and reveal a progressive, denoising-like pattern (Sec. 3.2).

Importantly, the depthwise trajectory depends on the transport task the model is asked to solve at inference. This becomes especially clear when the model generates a noisy intermediate $\mathbf{x}_r$ rather than the clean endpoint $\mathbf{x}_0$. Distilled one-step models under multi-step sampling nest depth as time within sampling time, with each sampler step carrying its own depthwise refinement (Sec. 4.1). Flow

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.

maps conditioned on the transport endpoint such as MeanFlow (Geng et al., 2025) and Shortcut models (Frans et al., 2025) instead exhibit a non-monotonic *denoise-then-renoise* pattern, first revealing cleaner image structure before renoising toward the requested noisy endpoint (Sec. 4.2).

Finally, in Sec. 5, we treat depth as time explicitly by training a single time-conditioned block to denoise across layers, yielding an even more compressed one-step generator. For example, we compress a MeanFlow SiT-L/2 one-step model for ImageNet-256 $\times$ 256 generation by $16.6\times$ (from 459M to 27.6M parameters) at FID 11.7, which improves to 4.9 with FD-loss post-training (Yang et al., 2026), compared with 4.0 for the original model. Performing the same compression on a drifting model, by contrast, degrades it to a FID of 47.1.

2 Preliminaries: One-step Generative Models

Diffusion- or flow-based generative models transport a simple noise distribution toward the data distribution along a time-indexed stochastic or deterministic process (Sohl-Dickstein et al., 2015; Ho et al., 2020; Liu et al., 2022; Lipman et al., 2022; Albergo et al., 2023). In the ordinary differential equation (ODE) formulation, the learned vector field specifies the instantaneous velocity with which samples evolve along this transport, and sampling amounts to numerically integrating it, conventionally requiring repeated network evaluations. A long-standing goal has therefore been to approximate this transport using only a few network evaluations—or even a single one—while preserving sample quality. Existing approaches broadly include distillation-based methods, which transfer the behavior of a multi-step teacher to a faster student through flow-map or distribution-level objectives (Salimans and Ho, 2022; Luo et al., 2023; Yin et al., 2024; Zhou et al., 2024; Sauer et al., 2024a;b), and methods that learn few-step models (or flow maps) from scratch (Song et al., 2023; Frans et al., 2025; Boffi et al., 2025; Geng et al., 2025; Zhou et al., 2025).

Flow maps (Boffi et al., 2025), in particular, provide a useful description of recent one-step generative models based on diffusion. A flow map $\Phi_{t\to r}$ denotes the transport induced by integrating the ODE from time t to time r. Throughout, we use transport task to refer to the specific map $\Phi_{t\to r}$ that a network is trained to approximate, i.e., the samplewise correspondence between noise level t and endpoint r that its output must realize; this is distinct from the downstream task (e.g., class-conditional generation) that the model serves. Learned flow maps can be classified by their transport task: approximating $\Phi_{t\to 0}(\mathbf{x}_t)$ for Song et al. (2023) and Geng et al. (2024), $\Phi_{t\to r}(\mathbf{x}_t)$ for Kim et al. (2024), Geng et al. (2025), and Zhou et al. (2025), and $\Phi_{t\to t-d}(\mathbf{x}_t)$ for Frans et al. (2025). The corresponding task conditioning $\boldsymbol{\tau}_{\text{task}}$ can thus be boundary-anchored ($t \to 0$), interval-conditioned ($t \to r$), or step-conditioned ($t \to t-d$), where the latter two are alternative parameterizations related by $r = t - d$.

For simplicity, we denote the $\boldsymbol{\tau}_{\text{task}}$-indexed network as follows, where its output may parameterize the transport map directly or through a velocity or displacement:

$$f_\theta(\mathbf{x}_t; \boldsymbol{\tau}_{\text{task}}) \quad \text{where} \quad \boldsymbol{\tau}_{\text{task}} = \begin{cases} t, & \textit{e.g.}\ \text{consistency models (Song et al., 2023)}, \\ (t, r), & \textit{e.g.}\ \text{MeanFlow (Geng et al., 2025)}, \\ (t, d), & \textit{e.g.}\ \text{Shortcut Models (Frans et al., 2025)}. \end{cases}$$

Here, $\mathbf{x}_t$ denotes the network input, t the current noise level, r the desired endpoint time of the transport, and d the desired step size. Standard diffusion networks are also conditioned on t, although this conditioning alone does not specify a boundary-anchored transport map.

We contrast these models with methods that lack an explicit time-indexed transport map at inference, e.g., drifting models (Deng et al., 2026), whose objectives constrain the distribution of generated samples without any time-indexed transport task. We use them as negative controls to distinguish generic layerwise refinement from computation organized by a $\boldsymbol{\tau}_{\text{task}}$-indexed transport task. We provide additional details on one-step generation in Appendix D.

3 Depth as Time

We introduce *depth as time*, an empirical observation in which denoising-like structure unfolds across the layers of a one-step generative model. We begin by introducing our methodology for comparing the intermediate temporal outputs of a multi-step flow model with the intermediate layer-wise outputs

of a one-step compressed model (Sec. 3.1). We then use this methodology to demonstrate the first evidence of progressive denoising across layers (Sec. 3.2), drawing three conclusions. First, the one-step layers exhibit patterns of denoising, which we call *depth as time*. Second, the recovered layer-wise mixing schedule in one-step models is different from the standard linear schedule in the temporal progression of multi-step models. Third, we demonstrate that drifting (τ-independent) models do not appear to exhibit depth as time unlike τ_{task}-conditioned models such as MeanFlows.

3.1 From sampling time to network depth

Multi-step flow models expose intermediate states across sampling time, whereas one-step generative models expose intermediate representations only across network depth. We therefore introduce a methodology for comparing these two forms of progression.

Consider a multi-step flow model that transports a noisy sample $\mathbf{x}_t$ toward a clean sample $\mathbf{x}_0$ by following a learned velocity field. Flow matching (Lipman et al., 2022; Albergo et al., 2023) defines a probability path between data and noise, *e.g.*, $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \sigma_t \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $t \in [0, 1]$. Under the linear interpolation considered here, $\alpha_t = 1 - t$ and $\sigma_t = t$, such that $t = 0$ corresponds to data and $t = 1$ corresponds to noise. Each sampled interpolation has velocity $\boldsymbol{\epsilon} - \mathbf{x}_0$, and the corresponding marginal velocity field is $\boldsymbol{u}_t(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{\epsilon} - \mathbf{x}_0 \mid \mathbf{x}_t = \boldsymbol{x}]$. The model is trained to approximate this field with $\boldsymbol{v}_\theta(\mathbf{x}_t, t) \approx \boldsymbol{u}_t(\mathbf{x}_t)$. Sampling then follows the ODE $d\mathbf{x}_t/dt = \boldsymbol{v}_\theta(\mathbf{x}_t, t)$, and exposes an explicit sequence of states $\mathbf{x}_{t_0}, \mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_K}$ across sampling time.

A one-step generator, by contrast, produces its output in a single network call and exposes no such external trajectory. It instead computes a sequence of hidden states $\mathbf{h}$ across network depth. Concretely, starting from a noisy input $\mathbf{z} = \mathbf{x}_t$ from time step t, and leveraging the task-conditioning variable $\boldsymbol{\tau}_{\text{task}}$ and class conditioning c, a one-step generator with a dimension-preserving residual stream produces:

$$\mathbf{h}_0 = \mathbf{E}(\mathbf{z}), \qquad \mathbf{h}_{i+1} = \mathbf{B}_{i+1}\left(\mathbf{h}_i; \boldsymbol{\tau}_{\text{task}}, c\right), \qquad i = 0, \ldots, L-1, \tag{1}$$

where $\mathbf{E}$ is the input embedding, $\mathbf{B}_{i+1}$ denotes the $(i+1)$-th network block, and L is the total number of blocks. Equivalently, the representation at depth i is obtained by composing the first i blocks, $\mathbf{h}_i = (\mathbf{B}_i \circ \cdots \circ \mathbf{B}_1)(\mathbf{h}_0)$, with the same conditioning supplied to each block. For a velocity-parameterized generator, the model's final prediction is $\boldsymbol{v}_\theta(\mathbf{z}, \boldsymbol{\tau}_{\text{task}}, c) = \mathbf{W}_{\text{out}}(\mathbf{h}_L; \boldsymbol{\tau}_{\text{task}}, c)$, where $\mathbf{W}_{\text{out}}$ denotes the network head, which includes normalization, adaptive layer normalization modulation (AdaLN), and a linear projection. Depending on the model, this output represents either an instantaneous velocity or an interval-averaged velocity.

Probe. To probe how the model's prediction develops across depth, we apply the same final head to the hidden state after each block, with the original conditioning and without executing the remaining blocks $\mathbf{B}_{i+1}, \ldots, \mathbf{B}_L$. For layer i, we define

$$\boldsymbol{v}_i \coloneqq \mathbf{W}_{\text{out}}\left(\mathbf{h}_i; \boldsymbol{\tau}_{\text{task}}, c\right). \tag{2}$$

At $i = L$, this recovers the model's ordinary output, $\boldsymbol{v}_L = \boldsymbol{v}_\theta(\mathbf{z}, \boldsymbol{\tau}_{\text{task}}, c)$. At intermediate layers, $\boldsymbol{v}_i$ measures the velocity prediction accessible through the output head from the residual stream at depth i. This probe introduces no learned parameters and does not modify the generator.[1]

We then convert each layerwise velocity estimate into the corresponding endpoint prediction. For the linear flow-matching interpolation, the clean-image estimate is $\hat{\mathbf{x}}_0 = \mathbf{x}_t - t\,\boldsymbol{v}_\theta(\mathbf{x}_t, t)$; with the exact marginal velocity field, which recovers the posterior mean $\mathbb{E}[\mathbf{x}_0 \mid \mathbf{x}_t]$. More generally, a velocity-parameterized one-step model predicts an endpoint at time r through $\hat{\mathbf{x}}_r = \mathbf{x}_t - (t - r)\,\boldsymbol{v}_\theta(\mathbf{x}_t, \boldsymbol{\tau}_{\text{task}}, c)$. For an instantaneous velocity prediction, this is a single Euler step, rather than an exact integration of the ODE. For an interval-averaged velocity prediction, it is the model's finite-interval transport parameterization as in Geng et al. (2025).

Applying this conversion to the layerwise readouts for a one-step input $\mathbf{x}_t$ gives our layerwise endpoint probe,

$$\tilde{\mathbf{x}}_r^i = \mathbf{x}_t - (t - r)\,\boldsymbol{v}_i. \tag{3}$$

[1] This parameter-free layerwise readout is closely related to the *logit lens* in language modeling (nostalgebraist, 2020). In Appendix B.1 we demonstrate that similar conclusions can be obtained using a *separately* trained linear probe, suggesting a lack of reliance on the particular form of the probe, but not by a *random* untrained probe. Therefore, the observed convergence is semantically meaningful rather than simply numerical.

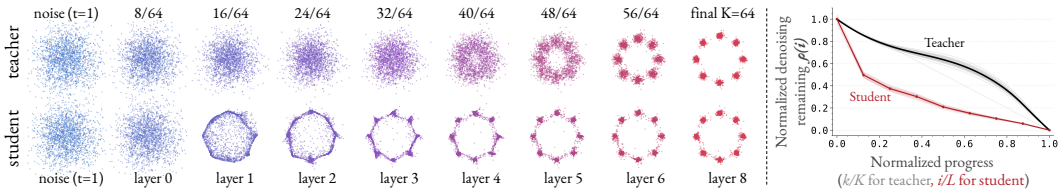


Figure 1: **A one-step toy student distilled from a multi-step teacher appears to denoise across depth on the 8-Gaussians toy example.** Normalized distance (right) shows that, although both progress toward the same endpoint, the student's depthwise progression need not trace the teacher's sampling-time trajectory. k/K is the normalized sampling step for the teacher, here $K = 64$ steps.

Here, r specifies the endpoint being decoded: $r = 0$ gives the clean-image predictions $\tilde{\mathbf{x}}_0^i$, while $0 < r < t$ gives a noisy intermediate predictions $\tilde{\mathbf{x}}_r^i$ at different depths i. At the final layer, $\tilde{\mathbf{x}}_r^L$ is the endpoint predicted by the full network for the corresponding transport problem. Simply put, $\tilde{\mathbf{x}}_r^i$ asks what endpoint prediction is already implied by the partially formed residual representation $\mathbf{h}_i$ at depth i. Interestingly, these layerwise velocities $\boldsymbol{v}_i$ used to construct $\tilde{\mathbf{x}}_r^i$ have conceptual connections to Neural ODE views of neural networks (Chen et al., 2018) and deep equilibrium models (Bai et al., 2019); we discuss these connections in Appendix A.

3.2 Evidence of progressive denoising across layers

Using the probe methodology of Sec. 3.1, we begin by demonstrating the emergence of depth-as-time in simple toy examples. Specifically, after progressively distilling (Salimans and Ho, 2022) a multi-step 8-Gaussians teacher into a one-step student, we plot the student's layerwise endpoint predictions $\tilde{\mathbf{x}}_0^i$; for the teacher, we plot $\hat{\mathbf{x}}_0$-predictions for each step along its multi-step sampling trajectory in Figure 1. The teacher $\mathbf{x}_0$-predictions are external and therefore different from internal layerwise $\tilde{\mathbf{x}}_0^i$ predictions. We also plot the normalized distance to the endpoint using Equation (4).

The toy example in Figure 1 exhibits layerwise refinement suggestive of denoising, so we next ask whether this phenomenon extends to large-scale one-step generators. We apply Equation (2) to recent one-step generative models trained with different objectives, including distillation (specifically, latent adversarial distillation for FLUX-schnell (Black Forest Labs et al., 2025)), MeanFlow (Geng et al., 2025), and Drifting (Deng et al., 2026), and visualize the resulting layerwise predictions $\tilde{\mathbf{x}}_0^i$ in Figure 2. For MeanFlow and Drifting, we use the released `B/2` checkpoints (12 layers, width 768) trained on ImageNet-256 $\times$ 256 (Russakovsky et al., 2015).

Qualitatively, FLUX-schnell and MeanFlow (both τ_{task}-conditioned models; Sec. 2) exhibit progressive refinement across depth. This progression is also reflected quantitatively by the approximately monotonic decrease in normalized endpoint distance from Equation (4). In contrast, the Drifting model does not exhibit the same systematic depthwise refinement.

In addition to plotting the constructed $\tilde{\mathbf{x}}_0^i$ predictions, we also quantify layerwise refinement, by measuring how far each intermediate prediction $\tilde{\mathbf{x}}_0^i$ at layer i (of the clean image $\mathbf{x}_0$) is from the model's final prediction $\tilde{\mathbf{x}}_0^L$. For a distance or dissimilarity measure d, we normalize this distance by its maximum along the same trajectory:

$$\boldsymbol{\rho}(i) := \frac{d(\tilde{\mathbf{x}}_0^i, \tilde{\mathbf{x}}_0^L)}{\max_{1 \le j \le L} d(\tilde{\mathbf{x}}_0^j, \tilde{\mathbf{x}}_0^L)}. \tag{4}$$

We normalize each trajectory separately before averaging across 32 examples, excluding trajectories for which all distances are zero. In our experiments, we use the ℓ_2 distance. Figures 1 and 2 show how distance evolves across intermediate layers in the toy example and for the one-step FLUX model along with their respective teachers.

Recovering an effective mixing schedule. We note that the layerwise refinement observed above, and in particular the normalized distance to the endpoint in Equation (4), is not by itself evidence of denoising. To test whether the layerwise decodes exhibit a denoising-like signal–noise structure, we take inspiration from the probability-path parameterization in Lipman et al. (2022) and solve the inverse problem of recovering depth-dependent mixing coefficients from the layerwise predictions in

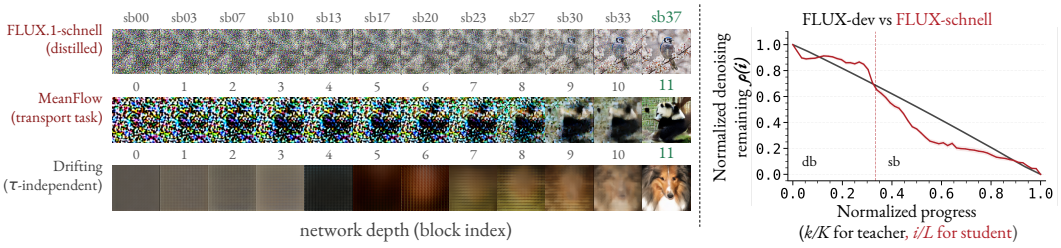


Figure 2: **Depthwise denoising-like progression persists in large-scale one-step image generators.** Layerwise endpoint predictions from FLUX-schnell and MeanFlow progressively reveal cleaner image structure across network depth, while Drifting provides a contrasting τ-independent model. For FLUX-schnell, normalized endpoint distance decreases similarly under the model's output head and an independently trained probe, whereas a random probe does not reproduce the same progression.

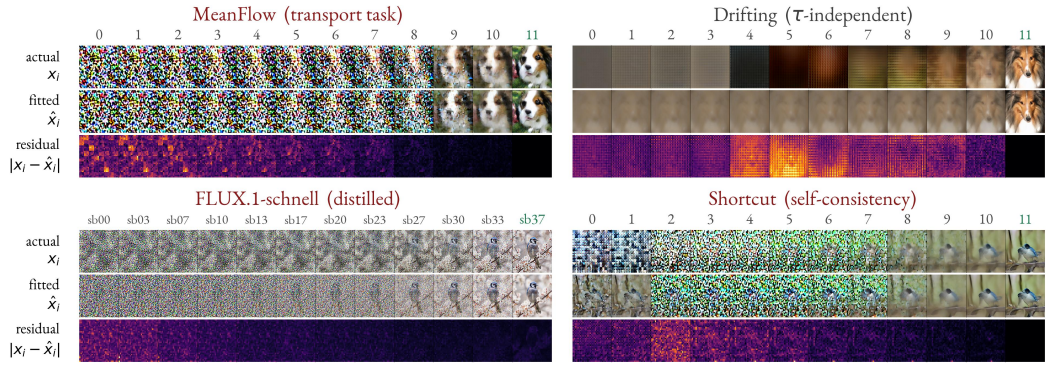


Figure 3: **Depthwise mixing across one-step generators.** A shared linear mixture of the input noise and final prediction closely reconstructs intermediate predictions for MeanFlow, FLUX-schnell, and Shortcut, with residuals decreasing across depth. In contrast, Drifting exhibits substantially larger residuals, indicating that the same two-component mixing model does not explain its depthwise trajectory as well.

Figure 2. The intuition is that, if a single pair of coefficients shared across trajectories can explain intermediate predictions as mixtures of the input noise and final prediction, then a denoising-like interpretation of depth is plausible.

We focus on full one-step generation from noisy input to clean output. We recover the coefficients for FLUX-schnell, a MeanFlow `B/2` model, and a Shortcut `B/2` model (Frans et al., 2025), and contrast these with a τ-independent Drifting `B/2` model. Specifically, for depthwise trajectory n of each model, we let $\boldsymbol{\epsilon}^n$ denote the input noise and $\tilde{\mathbf{x}}_0^{L,n}$ the final generated prediction, and assume

$$\tilde{\mathbf{x}}_0^{i,n} = \alpha_i \tilde{\mathbf{x}}_0^{L,n} + \sigma_i \boldsymbol{\epsilon}^n + \boldsymbol{\delta}_i^n, \tag{5}$$

where $\boldsymbol{\delta}_i^n$ is the unexplained residual. We fit a single pair of coefficients shared across 256 depthwise trajectories at each block: $(\widehat{\alpha}_i, \widehat{\sigma}_i) = \arg\min_{\alpha,\sigma} \sum_{n \in \mathcal{D}_{\text{train}}} \|\tilde{\mathbf{x}}_0^{i,n} - \alpha \tilde{\mathbf{x}}_0^{L,n} - \sigma \boldsymbol{\epsilon}^n\|_2^2$.

The mapping $i \mapsto (\widehat{\alpha}_i, \widehat{\sigma}_i)$ defines our *effective depthwise mixing schedule*. We impose neither monotonicity nor positivity, and we do not enforce the flow-matching constraint $\alpha_i + \sigma_i = 1$. Thus, the optimization is unconstrained and any observed difference in the signal and noise coefficients is completely attributed to the depth-as-time trajectories. We evaluate how well the fitted coefficients reconstruct decoded predictions on held-out trajectories in Figure 3.

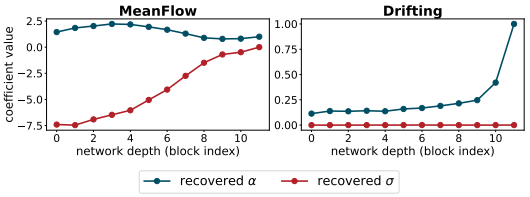


Figure 4: For Meanflows, the recovered depthwise schedule is *not* the flow-matching interpolation schedule. Drifting, on the other hand, maintains $\sigma_i \approx 0$ at every block.

Across the four models, Drifting stands out as having substantially larger reconstruction residuals between the observed and fitted depthwise

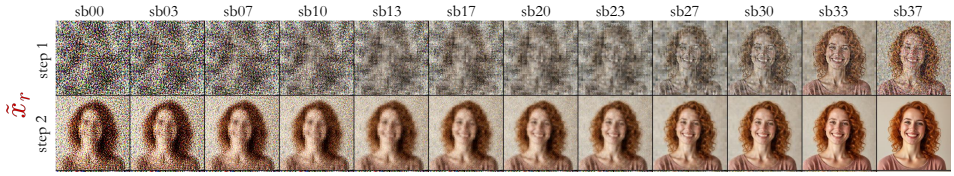


Figure 5: **Each sampler step exhibits its own depthwise denoising progression.** Layerwise endpoint predictions $\tilde{\mathbf{x}}_r^i$ for two-step FLUX-schnell, decoded after each single block (sb). Within each NFE, the prediction is refined across depth toward the endpoint prescribed by the sampler: a still-noisy intermediate state in step 1, and the clean image in step 2.

trajectories. In fact, its recovered noise coefficient is $\sigma_i \approx 0$ at every block (Figure 4), so its intermediate predictions carry no identifiable component of the input noise and do not follow a denoising-like path. In contrast, MeanFlow, FLUX-schnell, and Shortcut are each well described by a two-component signal–noise mixing model despite being trained with different objectives.

Notably, the recovered coefficients do not correspond to a standard flow-matching interpolant, for which $\alpha_i, \sigma_i \in [0, 1]$ and $\alpha_i + \sigma_i = 1$ along the path. For MeanFlow, for example, we recover $\alpha_4 = 2.18$ and $\sigma_4 = -6.04$ at block 4 (Figure 4), with both coefficients outside $[0, 1]$. Thus, although the low held-out reconstruction error for MeanFlow (≈ 0.10 on average across blocks) indicates a recoverable signal–noise organization across depth, the recovered schedule is distinct from the interpolation schedule used during flow-matching training. We interpret this as depth defining its own effective denoising-like coordinate, rather than directly reproducing training time. We provide additional details and recovered schedules for the other models in Appendix B.2.

4 Depth As Time Generalizes to Arbitrary Transport Tasks

Our findings so far show depth as time unfolding across the blocks of a single one-step evaluation that solves the full noise-to-clean task. However, one-step models based on diffusion are trained on a more general transport task that also supports multi-step generation. In this section, we show that depth as time extends to arbitrary start and end points along the generation trajectory, in two settings: distilled models in Sec. 4.1 and MeanFlow in Sec. 4.2 (and relegate Shortcut models to Appendix E.1).

In distilled models, we find that each sampling step contains its own depthwise refinement, nesting depth as time within sampling time. In MeanFlow, the depthwise trajectory adapts to the requested transport interval with an interesting property: when the prescribed endpoint is noisy, intermediate layers first denoise to reveal cleaner structure and then gradually *renoise* toward that endpoint.

4.1 Distilled one-step generators nest depth as time in sampling time

We first consider distilled one-step models, which are conditioned only on the current state and time $f_\theta(\mathbf{x}_t, t)$. Because they do not explicitly receive an endpoint r, multi-step generation relies on an external sampler that uses each network function evaluation (NFE) to advance from the current time t to a prescribed next time r. We refer to the resulting transport problem $t \to r$ as a *subtask* of the full $1 \to 0$ generation task.

To probe how computation unfolds within these subtasks, we study two large-scale distilled text-to-image models, FLUX-schnell and SANA-Sprint 0.6B (Chen et al., 2025). For each subtask, we decode both the endpoint prediction $\tilde{\mathbf{x}}_r^i$ and the diagnostic clean prediction $\tilde{\mathbf{x}}_0^i$ following Equation (3). The former shows how the layerwise prediction evolves toward the next state prescribed by the sampler; the latter shows what clean image is implied by the same intermediate representation.

Figure 5 shows the layerwise endpoint predictions for FLUX-schnell with generation split into two subtasks, and Figure 6 quantifies both predictions using Equation (4) for FLUX-schnell and SANA-Sprint across four subtasks.

Each subtask exhibits its own depthwise refinement. In Figure 5, the prediction in step 1 sharpens toward a still-noisy intermediate state; step 2 then restarts from that state and refines toward the clean

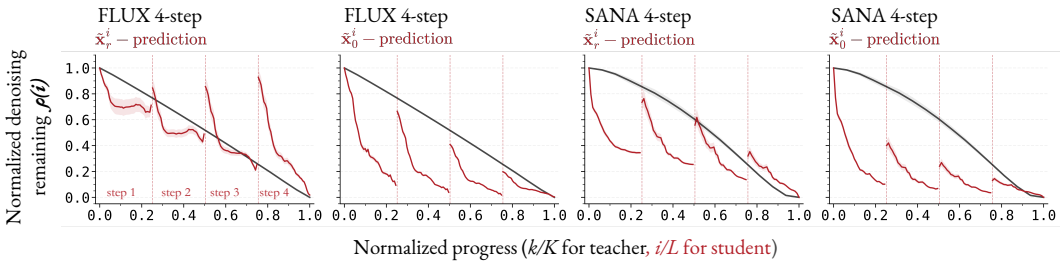
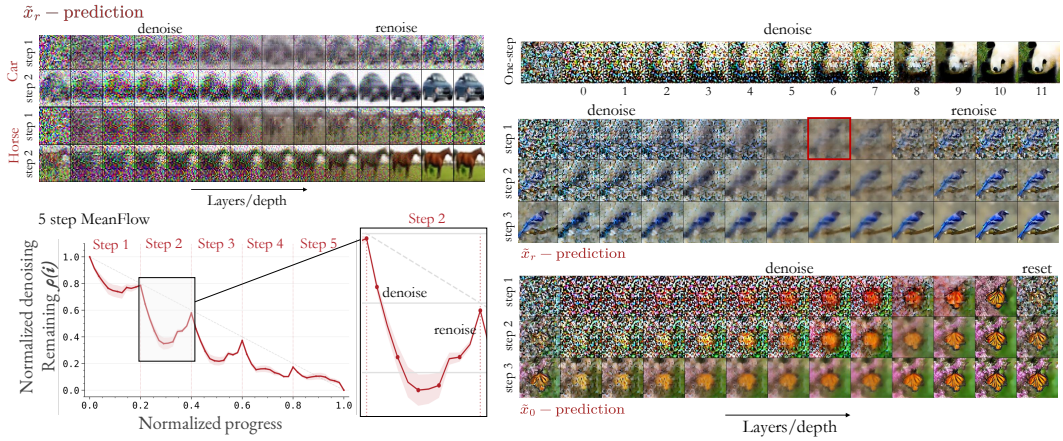


Figure 6: **Sampling time and network depth encode distinct but nested progressions.** Normalized distance $\rho(i)$ (Equation (4)) for four-step FLUX-schnell and SANA-Sprint, computed from the endpoint predictions $\tilde{\mathbf{x}}_r^i$ and the diagnostic clean predictions $\tilde{\mathbf{x}}_0^i$. Red curves show the layerwise progression within each NFE, gray curves show the multi-step teacher's sampling trajectory, and dashed lines mark sampler steps. The $\tilde{\mathbf{x}}_r^i$ curves reset at each step as the sampler advances, whereas the $\tilde{\mathbf{x}}_0^i$ curves largely reach the final clean image within the first step.



(a) MeanFlow on CIFAR-10 (DiT-B/2)

(b) MeanFlow on ImageNet (SiT-B/2)

Figure 7: **Multi-step MeanFlow models trained on CIFAR-10 and ImageNet exhibit the *denoise-then-renoise* pattern. (a)** Visualizing $\tilde{\mathbf{x}}_r$-predictions on the two-step generation trajectory across depth for a MeanFlow trained on CIFAR-10[2]reveals the denoise-then-renoise pattern (*top*). Quantitatively, the recovered depth schedule $\boldsymbol{\rho}(i)$ for a five-step generation trajectory , which decreases and then increases within each step (*bottom*). **(b)** We observe the depth as time on a MeanFlow with SiT-B/2 trained on ImageNet with one step generation (*top*). Using the same model but on three-step generation, we observe the same denoise and renoise pattern with $\tilde{\mathbf{x}}_r$-predictions (*middle*) and $\tilde{\mathbf{x}}_0$-predictions (*bottom*).

image. In Figure 6, this appears as the $\tilde{\mathbf{x}}_r^i$ curves jumping back up each time the sampler advances. This reset, however, applies only to the sampler state. The diagnostic $\tilde{\mathbf{x}}_0^i$ predictions show that even during the first subtask, $1 \to 0.75$, the final layerwise representation already largely implies the clean image at $t = 0$ (Figure 6, second and fourth panels). The amount of refinement in $\tilde{\mathbf{x}}_0^i$ also shrinks in later subtasks as the sampler itself approaches the clean endpoint.

The endpoint predictions $\tilde{\mathbf{x}}_r^i$, by contrast, do not progress as far toward the clean image within each step and can flatten or even reverse within depth, most visibly for FLUX (Figure 6, first panel). This is consistent with the model returning toward the prescribed noisy endpoint. In general, this suggests that sampling time and network depth are distinct but nested axes of progression. We examine the reversal in $\tilde{\mathbf{x}}_r^i$ more directly in explicitly interval-conditioned MeanFlow models in Sec. 4.2.

4.2 MeanFlows Denoise, Then Renoise, Toward Noisy Endpoints

[2]Most CIFAR-10 diffusion and one-step generation codebases use DDPM++/NCSN++-style U-Nets (Song et al., 2021); we train our own DiT-B/2 on CIFAR-10 for this experiment.

Unlike the distilled models of Sec. 4.1, which receive the endpoint r only implicitly through the external sampler, MeanFlow (Geng et al., 2025) is explicitly conditioned on the interval (t, r) and predicts the average velocity u over it. This lets us ask directly how the requested endpoint shapes the depthwise trajectory. The probe of Sec. 4.1 carries over directly, except that the final output head now estimates the interval-averaged velocity $\boldsymbol{u}_i$ not an instantaneous one. Decoding it gives the layerwise endpoint prediction $\tilde{\mathbf{x}}_r^i = \mathbf{z}_t - (t-r)\boldsymbol{u}_i$ and, for $r > 0$, a diagnostic clean-image estimate from the same velocity, $\tilde{\mathbf{x}}_0^i = \mathbf{z}_t - t\boldsymbol{u}_i$.

For one-step generation, the requested endpoint is clean ($r = 0$), and MeanFlow behaves as in Sec. 3.2: on ImageNet (Russakovsky et al., 2015), the layerwise predictions progress approximately monotonically from noise toward the clean image (Figure 7b, top row). Under multi-step sampling, each network evaluation instead solves a subtask $t \to r$ with $r > 0$. The diagnostic clean predictions $\tilde{\mathbf{x}}_0^i$ show that depth as time persists within each subtask: at every step, the layerwise predictions again progress toward the clean image, nesting depthwise refinement within sampling time as in the distilled models of Sec. 4.1 (Figure 7b, bottom rows).

The endpoint predictions $\tilde{\mathbf{x}}_r^i$, which track what the network *actually* outputs for each subtask, reveal a different pattern. Because the prescribed endpoint is now noisy, the depthwise trajectory becomes non-monotonic: intermediate layers first reveal a cleaner image structure than the desired target, and later layers gradually *renoise* the prediction toward $\mathbf{z}_r$. We observe this on both CIFAR-10 (Krizhevsky, 2009) and ImageNet (Figure 7a, top; Figure 7b, middle rows), where the cleanest prediction appears in the intermediate layers (highlighted in red) rather than the noisy endpoint.

The normalized distance $\rho(i)$ for a 5-step MeanFlow on CIFAR-10 quantifies the same effect: within each step, $\rho(i)$ first falls and then rises again before the next step begins (Figure 7a, bottom). We refer to this pattern as *denoise-then-renoise*. It suggests that even when asked for only a partial transport, the network passes through an estimate of the clean image.

Compositional hypothesis for *denoise-then-renoise*. At first glance, this trajectory is surprising. An interval-conditioned model is asked to solve the direct transport $\mathbf{z}_t \mapsto \mathbf{z}_r$, yet its intermediate predictions detour through the clean endpoint estimate before returning to the requested noisy one. One candidate explanation is that flow maps satisfy a *compositional consistency* property (Boffi et al., 2025). Writing $\Phi_{t\to r}$ for the transport from time t to time r, any intermediate time s gives $\Phi_{t\to r} = \Phi_{s\to r} \circ \Phi_{t\to s}$. Composing through the clean boundary, in particular, yields $\Phi_{t\to r}(\mathbf{z}_t) = \Phi_{0\to r}(\Phi_{t\to 0}(\mathbf{z}_t))$, a factorization $\mathbf{z}_t \to \mathbf{z}_0 \to \mathbf{z}_r$ that matches the observed trajectory. Yet, compositional consistency alone cannot be the full explanation since it holds equally for every intermediate time s and therefore does not single out the clean endpoint.

The clean endpoint estimate is instead singled out by the structure of the training target. For example, under the linear path $\mathbf{z}_t = (1-t)\mathbf{x}_0 + t\epsilon$, the marginal velocity is affine in the posterior clean estimate: $v(\mathbf{z}_t, t) = \mathbb{E}[\epsilon - \mathbf{x}_0 \mid \mathbf{z}_t] = \frac{\mathbf{z}_t - \mathbb{E}[\mathbf{x}_0 \mid \mathbf{z}_t]}{t}$ (Lipman et al., 2022; Albergo et al., 2023)[3]. A single Euler step to any endpoint r therefore gives $\mathbf{z}_t - (t-r)\, v(\mathbf{z}_t, t) = \frac{r}{t}\mathbf{z}_t + \left(1 - \frac{r}{t}\right)\mathbb{E}[\mathbf{x}_0 \mid \mathbf{z}_t]$, so every endpoint estimate with $r < t$ is an affine combination of $\mathbf{z}_t$ and $\mathbb{E}[\mathbf{x}_0 \mid \mathbf{z}_t]$. The posterior clean estimate is thus a non-trivial computation shared across all of these tasks.

For MeanFlow, this extends beyond a single Euler step: the average velocity $u(\mathbf{z}_t, t, r) = \frac{1}{t-r}\int_r^t v(\mathbf{z}_s, s)\, ds$ satisfies the MeanFlow identity: $u(\mathbf{z}_t, t, r) = v(\mathbf{z}_t, t) - (t-r)\,\frac{d}{dt}u(\mathbf{z}_t, t, r)$ (Geng et al., 2025). The first term, $v(\mathbf{z}_t, t)$, is independent of r and, determined by the clean estimate $\mathbb{E}[\mathbf{x}_0 \mid \mathbf{z}_t]$; the second is an interval-dependent correction that vanishes as $r \to t$. Therefore, a network trained jointly over many (t, r) pairs has a natural computation to reuse: first resolve the clean sample implied by $\mathbf{z}_t$, then apply the r-specific correction that carries it to the requested endpoint. This two-stage computation matches the observed denoise-then-renoise trajectory in Figure 7.

5 Making Depth as Time Explicit

So far we demonstrated that task-conditioned one-step models refine their predictions across layers in a denoising-like manner. A natural next question is whether this layerwise computation can itself be explicitly modeled as a *flow*, with a single weight-shared block applied repeatedly and conditioned

[3]The same holds for any affine path $\mathbf{z}_t = \alpha_t \mathbf{x}_0 + \sigma_t \epsilon$, since $\mathbb{E}[\epsilon \mid \mathbf{z}_t] = \big(\mathbf{z}_t - \alpha_t \mathbb{E}[\mathbf{x}_0 \mid \mathbf{z}_t]\big)/\sigma_t$.

on a time variable that loosely corresponds to the layer index. If so, the one-step model could be compressed further in its number of parameters, while also gaining the flexibility to vary the number of *depthwise* steps rather than being tied to a fixed number of layers L.

Concretely, the one-step model has a sequence of layer-specific blocks $\{\mathbf{B}_i\}_{i=1}^{L}$, with hidden layers $\mathbf{h}_{i+1} = \mathbf{B}_{i+1}(\mathbf{h}_i; \boldsymbol{\tau}_{\text{task}}, c)$ (eq. (1)). We have demonstrated that these blocks loosely correspond to denoising via depth as time. Making this explicit, we can rewrite the model as:

$$\mathbf{h}_{\tau_{i+1}} = \mathbf{B}_\theta\left(\mathbf{h}_{\tau_i}, (\tau_i, \tau_{i+1}); \tau_{\text{task}}, c\right), \tag{6}$$

with the time interval conditioning (τ_i, τ_{i+1}) replacing the layer-specific transformations. Intuitively, we can view this sequence as a flow in the representation space: each application of $\mathbf{B}_\theta$ over the computational interval (τ_i, τ_{i+1}) transports the hidden state $\mathbf{h}_{\tau_i}$ to $\mathbf{h}_{\tau_{i+1}}$. If we set $\tau_i = i/L$, we maintain L steps corresponding to the L layers.

We train the parameters $\mathbf{B}_\theta$ using the supervision signal from the L hidden layers $\mathbf{h}_1, \ldots, \mathbf{h}_L$ of the one-step model. We embed the conditioning variables, including the interval (τ_i, τ_{i+1}), and inject them into the shared block through adaptive layer normalization (AdaLN) modulation. This training is data-free and uses only random noise $z \sim \mathcal{N}(0, I)$, sampled class labels, and the one-step teacher model representations. See Appendix C.1 for training details.

Parameter compression. Using this formulation, we compress the MeanFlow `SiT-B/2` and `SiT-L/2` one-step models for ImageNet 256×256 from 131M and 459M to 16.0M and 27.6M parameters, respectively, corresponding to 8.2× and 16.6× compression. The compressed models reach FID-50k 15.6 and 11.7, which improve to 8.2 and 4.9 with FD-loss post-training (Yang et al., 2026), compared with 6.2 and 4.0 for the original models. Using two unique blocks instead of one trades some compression for quality, reaching FID 5.84 with 26.6M parameters (4.9× compression) for `SiT-B/2` and 4.04 with 46.5M parameters (9.9×) for `SiT-L/2` after post-training. We ablate the effect of conditioning on the interval (τ_i, τ_{i+1}) in Appendix C.4 and show that on CIFAR-10, removing it from an otherwise identical weight-shared block more than doubles FID-50k.

Compute compression. Because $\mathbf{B}_\theta$ now defines a flow over depth, we can also borrow intuition from accelerating diffusion models to reduce inference-time compute, training over coarse intervals (τ_a, τ_b) that span multiple layers so that $\mathbf{B}_\theta$ runs fewer than L times. For `SiT-L/2` with two unique blocks, halving the number of block applications gives FID 5.25, and reducing it by 3× gives 6.11, against a teacher FID of 4.01 (after post-training). In contrast, simply running the native-depth student with fewer applications degrades FID (Appendix C.3). Much like sampling steps in diffusion, the number of depthwise steps thus becomes a knob that trades compute for quality within a single forward pass.

MeanFlow models are more compressible than drifting models. Finally, recall that drifting models, which lack a time-indexed transport task or training signal, do not exhibit depth as time (Sec. 3.2), even though their predictions still sharpen in later blocks. Consistent with this, making depth as time explicit fails for them: with an analogous procedure (Appendix C.5), the drifting `B/2` student reaches only FID-50k 47.09, about three times worse than the MeanFlow `SiT-B/2` student (15.61) at a similar compression ratio, despite distilling from a stronger teacher (FID 1.75). It therefore seems that refinement along depth is not enough to result in a compressible one-step generator, and that the time-indexed transport task is what organizes this refinement into a denoising process compact enough for a single time-conditioned block.

Full training details and results are in Appendix C.

6 Conclusion

We began by asking what happens to the denoising trajectory of diffusion when generation is compressed into a single forward pass. Through a series of experiments, we showed that one-step generative models exhibit a denoising-like progression across network depth, a phenomenon we call *depth as time*. Decoding intermediate representations further revealed that this progression reflects the transport task being solved, from approximately monotonic denoising toward clean endpoints to *denoise-then-renoise* when the prescribed endpoint remains noisy. Finally, we made this depthwise structure explicit and reveal that models with explicit time-indexed transport maps are more compressible. Leveraging this property with weight sharing and coarse depthwise steps, we can further compress both parameters and compute, resulting in a more compute efficient one-step model.

AI USE STATEMENT

In this work, we used generative AI tools (Claude Code and Cursor) to implement methods, specifically for writing and debugging experimental code. We have not used generative AI tools to propose or refine hypotheses, develop conceptual frameworks, formulate mathematical claims, design research methodology or experiments, or interpret results, and generating synthetic datasets, assisting with proofs, translation, dataset cleaning, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools (Claude Opus 5.5 and GPT-6 Sol) for feedback on the presentation and readability of earlier drafts, for formatting LaTeX tables and figures, and for cross-checking reported numbers against training and evaluation logs. We have reviewed all AI-assisted work: all AI-assisted code was reviewed and tested by the authors, all reported numbers were verified against the original evaluation logs, all citations were manually checked against the original papers, and all text in the final version was written by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023. 2, 3, 8

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, Zico Kolter, and Roger Grosse. Path independent equilibrium models can better exploit test-time computation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. 15

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019. 4, 15

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pages 4965–4987, 2024. 1

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025. 4

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *TMLR*, 2025. 1, 2, 8, 15

Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16185–16195, 2025. 6, 25

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. 4, 15

Kevin Clark and Priyank Jaini. Text-to-image diffusion models are zero shot classifiers. *Advances in Neural Information Processing Systems*, 36:58921–58937, 2023. 1, 15

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. 15

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026. 1, 2, 4, 15, 21

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023. 1

David J Field. Relations between the statistics of natural images and the response properties of cortical cells. *Journal of the Optical Society of America A*, 4(12):2379–2394, 1987. 27

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=OlzB6LnXcS. 1, 2, 5, 15, 26

Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley Osher, and Wotao Yin. Jfb: Jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6648–6656, 2022. doi: 10.1609/aaai.v36i6.20619. 15

Zhengyang Geng, Meng-Hao Guo, Hongxu Chen, Xia Li, Ke Wei, and Zhouchen Lin. Is attention better than matrix decomposition? In *International Conference on Learning Representations*, 2021a. 15

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. In *Advances in Neural Information Processing Systems*, volume 34, pages 24247–24260, 2021b. 15

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024. 2

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025. 1, 2, 3, 4, 8, 15, 25

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11398–11442. PMLR, 2023. 15

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 2014. 15

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 11984–11995, 2020. 15

Ziyu Guo, Renrui Zhang, Hongyu Li, Manyuan Zhang, Xinyan Chen, Sifan Wang, Yan Feng, Peng Pei, and Pheng-Ann Heng. Thinking-while-generating: Interleaving textual reasoning throughout visual generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26295–26305, 2026. 1

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 1

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33: 6840–6851, 2020. 2

Dongjun Kim, Chieh-Hsin Lai, WeiHsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. In *International Conference on Learning Representations*, volume 2024, pages 44493–44525, 2024. 2, 15, 25

Harsh Kohli, Srinivasan Parthasarathy, Huan Sun, and Yuekun Yao. Loop, think, & generalize: Implicit reasoning in recurrent-depth transformers, 2026. 15

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 8

Asher Labovich. Stability and generalization in looped transformers, 2026. 15

Chieh-Hsin Lai, Yang Song, Dongjun Kim, Yuki Mitsufuji, and Stefano Ermon. The principles of diffusion models. *arXiv preprint arXiv:2510.21890*, 2025. 25

Junsung Lee, Hyunsoo Lee, Yong Jae Lee, and Bohyung Han. Low-resolution editing is all you need for high-resolution editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16216–16225, 2026. 1

Alexander C Li, Mihir Prabhudesai, Shivam Duggal, Ellis Brown, and Deepak Pathak. Your diffusion model is secretly a zero-shot classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2206–2217, 2023. 1, 15

Xinran Liang, Esin Tureci, Prachi Sinha, Ye Zhu, Vikram V Ramaswamy, and Olga Russakovsky. Personalized generative models for contextual debiasing. *arXiv preprint arXiv:2605.26353*, 2026. 1

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 2, 3, 4, 8

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. 2

Yizhou Liu, Sara Kangaslahti, Ziming Liu, and Jeff Gore. Inverse depth scaling from most layers being similar, 2026. 15

Wenquan Lu, Yuechuan Yang, Kyle Lee, Yanshu Li, and Enqi Liu. Latent chain-of-thought? decoding the depth-recurrent transformer, 2025. 15

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-Instruct: A universal approach for transferring knowledge from pre-trained diffusion models. *NeurIPS*, 36:76525–76546, 2023. 1, 2, 15

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021. 1

nostalgebraist. Interpreting gpt: The logit lens. LessWrong, August 2020. Posted 31 August 2020. 3, 15

Francesco Pappone, Donato Crisostomi, and Emanuele Rodolà. Two-scale latent dynamics for recurrent-depth transformers. *arXiv preprint arXiv:2509.23314*, 2025. 15

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 4, 8, 26

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 1, 2, 4, 15

Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024a. 2, 15, 25

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pages 87–103. Springer, 2024b. 2, 15

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. 15

Kristian Schwethelm, Daniel Rueckert, and Georgios Kaissis. How much is one recurrence worth? iso-depth scaling laws for looped language models, 2026. 15

Makoto Shing, Masanori Koyama, and Takuya Akiba. Diffusionblocks: Block-wise neural network training via diffusion interpretation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pwVSmK71cS. 15

Eero P Simoncelli and Bruno A Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001. 27

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pages 2256–2265. pmlr, 2015. 2

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS. 7

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023. 1, 2, 15, 25

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8228–8238, 2024. 1

Hanyang Wang, Yiyang Liu, Jiawei Chi, Fangfu Liu, Ran Xue, and Yueqi Duan. Cfg-ctrl: Control-based classifier-free diffusion guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11437–11447, 2026. 1

Kevin Xu and Issei Sato. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. 15

Jiawei Yang, Zhengyang Geng, Xuan Ju, Yonglong Tian, and Yue Wang. Representation fr\'echet loss for visual generation. *arXiv preprint arXiv:2604.28190*, 2026. 2, 9, 18

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024. 15

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6613–6623, 2024. 1, 2, 15, 25

Arman Zarei, Samyadeep Basu, Mobina Pournemat, Sayan Nag, Ryan A Rossi, and Soheil Feizi. Slideredit: Continuous image editing with fine-grained instruction control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14430–14439, 2026. 1

Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. *arXiv preprint arXiv:2503.07565*, 2025. 2, 15

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity distillation: Exponentially fast distillation of pretrained diffusion models for one-step generation. In *Forty-first International Conference on Machine Learning*, 2024. 1, 2, 15

Appendix

A Related Work and Discussion . . . 15

B Readout Controls and Depthwise Mixing Details . . . 16

B.1 Readout controls . . . 16

B.2 Depthwise mixing details . . . 16

C Implementation Details for Making Depth as Time Explicit . . . 17

C.1 Distilling the depthwise trajectory . . . 17

C.2 FD-loss post-training . . . 18

C.3 Results . . . 19

C.4 Ablating the internal time conditioning . . . 20

C.5 Drifting models as a negative control . . . 21

D Extra Background on One-Step Generators . . . 25

D.1 Distillation . . . 25

D.2 Boundary-anchored and interval-conditioned flow maps . . . 25

D.3 Step-conditioned flow maps: Shortcut models . . . 26

E Extra Analyses on Flow Maps . . . 26

E.1 Shortcut Models . . . 26

A Related Work and Discussion

This appendix expands the related work in the main text. Multi-step generators compute across network calls in sampling time, while a one-step generator computes across the layers of a single forward pass. We ask how the former is reorganized into the latter, and whether the resulting depthwise trajectory can itself be modeled and compressed as a flow.

Depth as a dynamical process. Neural ODEs (Chen et al., 2018) view a residual update $\mathbf{h}_{i+1} = \mathbf{h}_i + g(\mathbf{h}_i)$ as an Euler step, with depth playing the role of time. Our layerwise predictions $\boldsymbol{v}_i = \mathbf{W}_{\text{out}}(\mathbf{h}_i; \boldsymbol{\tau}_{\text{task}}, c)$, obtained by applying the model's output layer (normalization, AdaLN modulation, and linear projection) to each intermediate state, make this view measurable in generative models. Universal Transformers (Dehghani et al., 2019) apply one shared block repeatedly with a per-step timestep embedding, the closest precedent for our explicit model (Section 5); our block is instead conditioned on an interval (τ_a, τ_b) of an internal clock, which allows steps of different sizes. DiffusionBlocks (Shing et al., 2026) treats residual blocks as denoising steps by design, assigning each block a range of noise levels so that blocks can be trained independently; we find that this correspondence already emerges in one-step generators trained end-to-end. Deep equilibrium models (Bai et al., 2019; Anil et al., 2022) and related implicit models (Geng et al., 2021a;b; Fung et al., 2022; Gu et al., 2020; Labovich, 2026) define the representation as a fixed point $\mathbf{z}^\star = f_\theta(\mathbf{z}^\star; \mathbf{x})$.

Layerwise predictions and probing. Computing layerwise predictions with the model's own output layer is the generative analogue of the *logit lens* (nostalgebraist, 2020), which decodes the intermediate states of language models with the unembedding and reveals a gradual refinement toward the final prediction. Because what such decodings reveal can depend strongly on the layer and on the choice of decoder (Lu et al., 2025), we confirm our results with a trained and a random linear probe (Appendix B.1). Representations of diffusion models have mostly been studied across sampling time (Li et al., 2023; Clark and Jaini, 2023); we study them across depth within a single evaluation.

Depth as iterative refinement. Depth often implements incremental refinement rather than distinct transformations (Liu et al., 2026; Pappone et al., 2025). We find that in one-step generators this refinement is shaped by the transport task: it follows a denoising-like schedule (Section 3.2), adapts to the requested endpoint (Section 4.2), and, unlike in drifting models, can be compressed into a single time-conditioned block (Section 5, Appendix C.5).

Weight-tied and looped computation. Weight tying decouples computation from parameters. Looped Transformers can emulate programmable computers (Giannou et al., 2023), learn iterative algorithms (Yang et al., 2024), match deeper models (Saunshi et al., 2025), and extrapolate to more iterations than seen in training (Kohli et al., 2026), though an extra iteration is worth less than an extra layer (Schwethelm et al., 2026). Timestep encodings recover iteration-dependent behavior in looped models (Xu and Sato, 2025), and our internal clock plays this role for generation (Appendix C.4).

One-step generative models. One-step generators are obtained by distilling a multi-step teacher (Salimans and Ho, 2022; Luo et al., 2023; Yin et al., 2024; Zhou et al., 2024; Sauer et al., 2024a;b) or by learning flow maps directly (Song et al., 2023; Kim et al., 2024; Frans et al., 2025; Boffi et al., 2025; Geng et al., 2025; Zhou et al., 2025), while GANs (Goodfellow et al., 2014) and drifting models (Deng et al., 2026) match only the output distribution (Appendix D). These methods compress sampling time; we ask where that computation goes, and show that it can be compressed further across depth.

B Readout Controls and Depthwise Mixing Details

B.1 Readout Controls

A natural concern is that the endpoint-convergence pattern measured by Eq. (4) may be an artifact of applying the model's final readout $\mathbf{W}_{\text{out}}$ to intermediate residual states. Since this readout is optimized only for the final representation $\mathbf{h}_L$, intermediate states might gradually become more compatible with it regardless of whether they encode anything meaningful. To assess how the pattern depends on the readout, we compare $\mathbf{W}_{\text{out}}$ with a trained and a random linear probe.

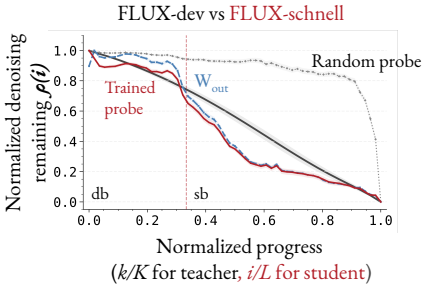


Figure 8: **Readout controls on one-step FLUX-schnell.** Normalized distance $\rho(i)$ to the final prediction under the model's own readout $\mathbf{W}_{\text{out}}$ (blue), a trained linear probe (red), and a random linear probe (grey), alongside the FLUX-dev teacher's sampling trajectory (black). The dashed line marks the transition from double-stream (db) to single-stream (sb) blocks.

Trained probe. We fit an independent linear map $\mathbf{V}$ by closed-form ridge regression on the layer-normalized final hidden state, such that $\mathbf{V}\big(\text{LayerNorm}(\mathbf{h}_L)\big) \approx \mathbf{W}_{\text{out}}\big(\mathbf{h}_L\big)$. The target is the model's own final prediction, which may be a velocity $\boldsymbol{v}$, noise $\boldsymbol{\epsilon}$, or image $\mathbf{x}$. We then freeze $\mathbf{V}$ and apply it to every intermediate hidden state $\mathbf{h}_i$; the probe is never trained on intermediate layers. On one-step FLUX-schnell, the trained probe traces nearly the same trajectory as $\mathbf{W}_{\text{out}}$ (Fig. 8), indicating that the pattern does not depend on the particular form of the readout.

Random probe. We also evaluate a frozen random linear map with the same input and output dimensions as $\mathbf{V}$. We sample $\mathbf{V}_{\text{rand}} \sim \mathcal{N}(0,1)^{D \times d_{\text{out}}}$ and rescale its weights to match the entrywise standard deviation of $\mathbf{V}$. At each layer, we compute $\boldsymbol{v}_i^{\text{rand}} = \mathbf{V}_{\text{rand}}\big(\text{LayerNorm}(\mathbf{h}_i)\big)$, without training the probe to match the model's predictions. The random probe does not reproduce the convergence pattern (Fig. 8): its $\rho(i)$ stays close to 1 through nearly all blocks and drops only in the last few, where it reaches zero simply because each trajectory is normalized to its own endpoint. The progressive convergence observed with $\mathbf{W}_{\text{out}}$ is therefore semantically meaningful rather than a numerical artifact of applying any linear map.

B.2 Depthwise Mixing Details

Mixing model. For trajectory n, let $\boldsymbol{\epsilon}^n$ be the Gaussian input latent and $\tilde{\mathbf{x}}_0^{L,n}$ the model's final endpoint prediction. At each block i, we fit one coefficient pair (α_i, σ_i), shared across trajectories, by unconstrained least squares over all spatial and latent dimensions:

$$\tilde{\mathbf{x}}_0^{i,n} \;\approx\; \alpha_i\, \tilde{\mathbf{x}}_0^{L,n} + \sigma_i\, \boldsymbol{\epsilon}^n.$$

Coefficients are fit on a training split, and for $i < L$ we report the held-out reconstruction error

$$R_i = \frac{\sum_n \big\|\tilde{\mathbf{x}}_0^{i,n} - \widehat{\alpha}_i\, \tilde{\mathbf{x}}_0^{L,n} - \widehat{\sigma}_i\, \boldsymbol{\epsilon}^n\big\|_2^2}{\sum_n \big\|\tilde{\mathbf{x}}_0^{i,n} - \tilde{\mathbf{x}}_0^{L,n}\big\|_2^2},$$

normalized by the error of predicting the final output directly, so that $R_i = 1$ means no improvement over this baseline.

Setup. We analyze MeanFlow `B/2` ($L = 12$), Drifting `B/2` ($L = 12$), FLUX-schnell (single-stream blocks, $L = 38$), SANA-Sprint ($L = 28$), and Shortcut `B/2` ($L = 12$). For each, we extract $N = 256$ trajectories from a single network evaluation with a fixed seed and split them $70/30$ into 179 training and 77 held-out trajectories.

Results. For the four task-conditioned models, σ_i rises from strongly negative values toward zero across depth while α_i remains of order one (Figure 9), a systematic signal–noise mixing structure. For MeanFlow, this structure is fit with low held-out error (mean $R \approx 0.10$), and its coefficients extend well outside the $[0, 1]$ range of a flow-matching interpolant, so depth follows its own effective schedule. For Drifting, $\sigma_i \approx 0$ at every block, so the input noise plays no identifiable role, and the fit is substantially worse (mean $R \approx 0.47$).

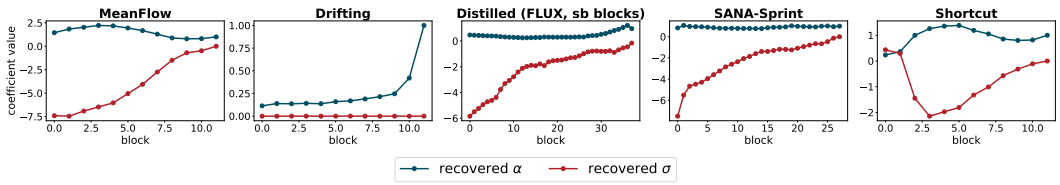


Figure 9: **Recovered depthwise mixing coefficients** (α_i, σ_i). For the task-conditioned models, σ_i rises toward zero across depth; for Drifting, $\sigma_i \approx 0$ at every block.

Normalization control. MeanFlow's readout uses LayerNorm, which centers its input, whereas Drifting's uses RMSNorm, which does not, so centering could in principle induce the mixing structure. We therefore swap only the normalization, keeping the trained AdaLN modulation and linear projection: we remove centering from MeanFlow and add it to Drifting, then refit the coefficients ($N = 192$, same protocol). Removing centering leaves MeanFlow's R_i essentially unchanged, and adding it to Drifting makes the fit marginally worse (Figure 10), so the contrast is not an artifact of normalization.

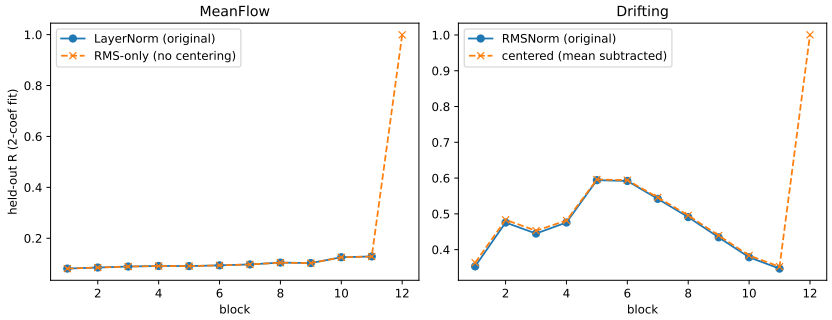


Figure 10: **Centering ablation.** For MeanFlow, removing centering leaves the held-out reconstruction error essentially unchanged across intermediate blocks. For Drifting, adding centering does not improve the fit and instead makes it marginally worse. The final block is excluded from the normalized R_i comparison because its baseline error is zero by construction.

C Implementation Details for Making Depth as Time Explicit

Here, we provide the details behind Section 5. We first describe how the weight-shared block $\mathbf{B}_\theta$ is trained by distilling the depthwise trajectory of a one-step teacher, and how it is trained and run with coarser steps at inference (App. C.1). We then describe FD-loss post-training (App. C.2) and report full results for parameter and compute compression, including qualitative samples (App. C.3). Finally, we ablate the internal clock on CIFAR-10 (App. C.4) and detail the drifting control (App. C.5).

C.1 Distilling the Depthwise Trajectory

Architecture. The student replaces the L layer-specific blocks of a one-step MeanFlow teacher (`SiT-B/2`, $L = 12$; `SiT-L/2`, $L = 24$) with $n \in \{1, 2\}$ unique blocks, applied in alternation for K steps. Each application is conditioned on its interval of the internal clock (τ_a, τ_b), together with the task conditioning $\boldsymbol{\tau}_{\text{task}} = (t, r) = (1, 0)$ for one-step generation and the class label c, all injected through AdaLN modulation. The input embedding $\mathbf{E}$ and output readout $\mathbf{W}_{\text{out}}$ are copied from the teacher and kept frozen, so only the shared blocks and the clock embedding are trained. The shared blocks are warm-started from the teacher's middle block.

Objective. Let $\{\mathbf{H}_j\}_{j=1}^{L}$ denote the teacher's hidden states for an input $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and label c. A student with K applications uses stride $s = L/K$, and the state $\mathbf{h}_k$ reached after its k-th application is trained to match the teacher state $\mathbf{H}_{ks}$. Because the norms of the teacher's hidden states grow with depth, an unnormalized loss would be dominated by the last layers and effectively ignore early ones, so we normalize each term by the norm of its teacher state. We additionally anchor the final decoded

prediction to the teacher's one-step sample $\mathbf{x}_0^T$:

$$\mathcal{L}_{\text{comp}}(\theta) = \mathbb{E}_{\mathbf{z},c}\left[\frac{1}{K}\sum_{k=1}^{K}\frac{\|\mathbf{h}_k - \mathbf{H}_{ks}\|_2^2}{\|\mathbf{H}_{ks}\|_2^2} + \lambda\left\|(\mathbf{z}-\boldsymbol{v}) - \mathbf{x}_0^T\right\|_2^2\right], \tag{7}$$

where $\boldsymbol{v} = \mathbf{W}_{\text{out}}(\mathbf{h}_K; \boldsymbol{\tau}_{\text{task}}, c)$ is decoded from the final student state and $\lambda = 1$. Since the teacher provides its full deterministic trajectory, no self-consistency objective is required, and training is data-free: it uses only noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and sampled class labels, which are replaced by the null label with probability $p_{\text{unc}} = 0.1$ to support classifier-free guidance (CFG).

Fixed-K and elastic training. A *fixed-K* student is trained for a single number of applications (Alg. C.1). An *elastic* student samples K uniformly from a set $\mathcal{K}$ at every iteration (Alg. C.2), so that one model can be run at any depth in the set; we use $\mathcal{K} = \{2, 4, 6, 12\}$ for `B/2` and $\mathcal{K} = \{4, 6, 8, 12, 24\}$ for `L/2`. The parameter-compression models in Section 5 are fixed-K students with $K = L$, so that $s = 1$ and every teacher state is supervised. When $K < L$, a few applications over coarse intervals replace several fine-grained transitions along depth (Fig. 11).

Hyperparameters. All fixed-K students, for both `B/2` and `L/2` and every number of unique blocks, are trained for 1.2M iterations with batch size 32, learning rate 10^{-4} with 2,000 warmup steps, and EMA decay 0.9999. The elastic students use the same settings.

Algorithm C.1 Fixed-K Training

req: teacher T (depth D), target K
$s \leftarrow D/K$ # fixed depth
repeat
  $z \sim \mathcal{N}(0, I)$, $y \sim p(y)$ # $y \to \varnothing$ w.p. p_{unc}
  $\{H_j\}_{j=1}^{D} \leftarrow T(z, y)$ # frozen
  $\tau_a \leftarrow \text{lin}(0, 1, K)$, $\tau_b \leftarrow [\tau_a^{2:}, 1]$
  $\{\mathbf{h}_k\}_{k=1}^{K}, v \leftarrow f_\theta(z, y; \tau_a, \tau_b)$
  $\mathcal{L}_{\mathbf{h}} \leftarrow \frac{1}{K}\sum_k \|\hat{\mathbf{h}}_k - \hat{H}_{ks}\|^2$
  $\mathcal{L}_e \leftarrow \|(z - v) - \mathbf{x}_0^T\|^2$
  $\theta \leftarrow \theta - \eta\,\nabla_\theta(\mathcal{L}_h + \lambda\,\mathcal{L}_e)$
until converged

Algorithm C.2 Elastic Training

req: teacher T (depth D), set $\mathcal{K}$
repeat
  $z \sim \mathcal{N}(0, I)$, $y \sim p(y)$ # $y \to \varnothing$ w.p. p_{unc}
  $\{H_j\}_{j=1}^{D} \leftarrow T(z, y)$ # frozen
  $K \sim \text{Unif}(\mathcal{K})$, $s \leftarrow D/K$
  $\tau_a \leftarrow \text{lin}(0, 1, K)$, $\tau_b \leftarrow [\tau_a^{2:}, 1]$
  $\{\mathbf{h}_k\}_{k=1}^{K}, v \leftarrow f_\theta(z, y; \tau_a, \tau_b)$
  $\mathcal{L}_{\mathbf{h}} \leftarrow \frac{1}{K}\sum_k \|\hat{\mathbf{h}}_k - \hat{H}_{ks}\|^2$
  $\mathcal{L}_e \leftarrow \|(z - v) - \mathbf{x}_0^T\|^2$
  $\theta \leftarrow \theta - \eta\,\nabla_\theta(\mathcal{L}_h + \lambda\,\mathcal{L}_e)$
until converged

Inference and compute accounting. Alg. C.3 generates a sample by running the student for K applications and decoding once with $\mathbf{W}_{\text{out}}$. With CFG, this is done twice. All FLOPs in Table 2 are reported relative to the same student at its native depth $K = L$, with CFG in both cases. Note that the MeanFlow teacher incorporates guidance into training and is evaluated with a single pass ($\omega = 1$), so a student at its native depth costs roughly twice the teacher per sample.

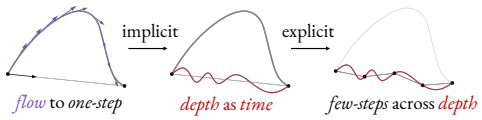


Figure 11: A single application of $\mathbf{B}_\theta$ over a coarse interval (τ_a, τ_b) replaces several fine-grained transitions.

C.2 FD-LOSS POST-TRAINING

After distillation, we optionally fine-tune each student with the FD loss of Yang et al. (2026), which backpropagates directly through the Fréchet distance between generated and real feature statistics rather than through a per-sample loss. We compute the distance jointly in three feature spaces, SigLIP, Inception, and MAE, each normalized so that no single space dominates, against precomputed moments of real ImageNet features. The population moments of generated features are tracked with an EMA of decay 0.999. We fine-tune for 20,000 steps with learning rate 10^{-6} and effective batch size 32 (4×8 gradient accumu-

Algorithm C.3 Inference (1 NFE Generation)

req: tied block b_θ, guidance ω
choose K # fixed: K; elastic: $K \in \mathcal{K}$
$z \sim \mathcal{N}(0, I)$, $y \sim p(y)$
$\tau_a \leftarrow \text{lin}(0, 1, K)$, $\tau_b \leftarrow [\tau_a^{2:}, 1]$
for $c \in \{y, \varnothing\}$: # cond/uncond
  $h \leftarrow \mathbf{E}(z)$
  for $k = 1..K$: $h \leftarrow b_\theta(h;\, \tau_a[k], \tau_b[k]; \tau_{\text{task}}, c)$
  $v_c \leftarrow \mathbf{W}_{\text{out}}(h)$ # decode once
$v \leftarrow v_\varnothing + \omega\,(v_y - v_\varnothing)$ # guidance
$\mathbf{x}_0 \leftarrow z - v$ # clean sample
return $\mathbf{x}_0$

lation), keeping the CFG scale fixed at the optimum of the distilled model. Unlike distillation, this stage uses statistics of real data.

C.3 Results

Parameter compression. Table 1 reports all fixed-K students at the teacher's depth. Adding unique blocks improves FID at the cost of less compression: for `B/2`, pre-FD FID improves from 15.61 with one block to 12.43 with two. FD-loss post-training improves every model by 3–7.5 FID, with larger gains for weaker starting models. Figure 12 places these models among existing one-step generators, and Figures 15–17 show paired samples from the `SiT-L/2` teacher and its students.

Model	Unique blocks	Params	K	Compression	FID (pre-FD)	FID (post-FD)	ω
MeanFlow `SiT-B/2` (teacher)	12	131M	12	–	6.17	–	–
Ours	1	16.0M	12	**8.2**×	15.61	8.17	1.4
Ours	2	26.6M	12	4.9×	12.43	5.84	1.2
MeanFlow `SiT-L/2` (teacher)	24	459M	24	–	4.01	–	–
Ours	1	27.6M	24	**16.6**×	11.75	4.87	1.4
Ours	2	46.5M	24	9.9×	7.19	4.04	1.3

Table 1: **Parameter compression on ImageNet** 256×256**.** FID-50k of fixed-K students at the teacher's depth, before and after FD-loss post-training, each at its optimal CFG scale ω. Compression is relative to the teacher's parameter count. Highlighted rows use a single shared block and are the results reported in Section 5; grey rows are the teachers. Adding unique blocks trades compression for quality.

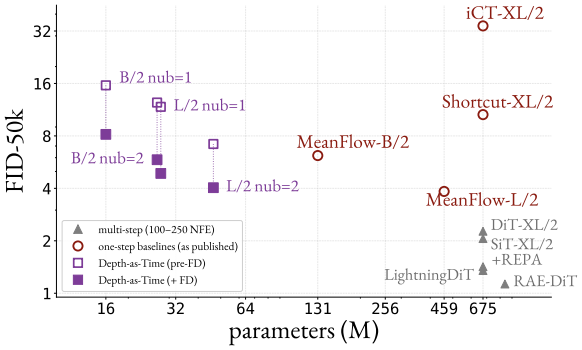


Figure 12: **Parameter efficiency of *depth as time* on ImageNet** 256×256**.** FID-50k is reported at each model's optimal CFG scale. Hollow markers denote models before FD-loss post-training; filled markers denote the same models after. All baselines are as reported in their original papers. Our students reach competitive FID with 4.9–16.6× fewer parameters than their teachers.

Compute compression. Table 2 reports FID after FD-loss post-training as the number of block applications decreases, for two-block students; Table 3 gives the corresponding pre-FD numbers for the elastic models, and Figure 13 plots both. Three observations stand out. First, a model trained at its native depth cannot simply be run with fewer, coarser applications: probing the native-depth model below its training depth collapses to FID above 290. Second, students trained for coarse intervals degrade gracefully, especially at larger scale: the `L/2` fixed-K student reaches FID 5.25 with half the block applications and 6.11 with a third, against a teacher FID of 4.01. Third, a single elastic model tracks the separately trained fixed-K models closely at moderate compression (*e.g.*, 6.13 vs. 6.11 for `L/2` at $K = 8$), but falls behind at the most aggressive setting.

`B/2` (teacher FID 6.17)

K	FLOPs ↓	elastic ↓	fixed-K ↓	native, probed ↓
12 (native)	1.00×	8.75	**5.84**	**5.84**
6	0.50×	10.17	**8.02**	294.68
4	0.33×	15.24	**11.31**	313.64
2	0.17×	48.01	**23.65**	436.60

`L/2` (teacher FID 4.01)

K	FLOPs ↓	elastic ↓	fixed-K ↓	native, probed ↓
24 (native)	1.00×	5.64	**4.04**	**4.04**
12	0.50×	5.81	**5.25**	474.65
8	0.33×	6.13	**6.11**	461.09
6	0.25×	7.69	**7.04**	613.43
4	0.17×	17.29	**10.27**	637.97

Table 2: **Compute compression on ImageNet** 256×256**.** FID-50k after FD-loss post-training for two-block students, each at its optimal CFG scale. FLOPs are relative to the same student at its native depth, with CFG in both cases. *Elastic* is a single model trained over the set $\mathcal{K}$ and run at each K. *Fixed-K* is a separate model trained for each K, and degrades gracefully as compute decreases. *Native, probed* is the native-depth model run with fewer, coarser applications than it was trained for, which collapses.

B/2 elastic, pre-FD

K	12	6	4	2
FID-50k	17.89	24.06	37.36	82.02
ω	1.5	1.5	1.5	1.7

L/2 elastic, pre-FD

K	24	12	8	6	4
FID-50k	10.55	12.33	15.91	24.23	47.56
ω	1.5	1.5	1.5	1.5	1.7

Table 3: **Pre-FD FID-50k of the elastic students** in Table 2, at the optimal CFG scale ω for each K.

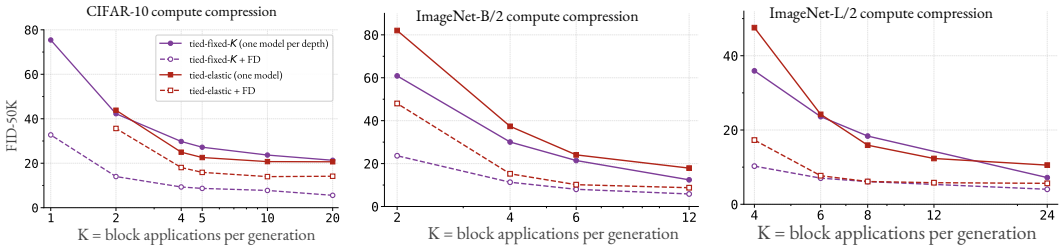


Figure 13: **Compute compression of depth as time.** FID-50k at the optimal CFG scale against the number of block applications K per generation; FLOPs scale approximately linearly with K. *Fixed-K* is a separate tied model trained for each depth; *elastic* is a single tied model run at every K. Solid markers are pre-FD, hollow markers post-FD.

C.4 Ablating the internal time conditioning

To test whether the benefit of weight sharing comes from treating depth as time, we compare two weight-tied models that differ only in whether they receive the internal clock. No publicly released DiT-based MeanFlow model exists for CIFAR-10 to distill from, so both are trained from scratch with the MeanFlow (t, r) objective; the CIFAR-10 MeanFlow model analyzed in Fig. 7 is likewise our own.

Setup. Both models use a `DiT-B/2` backbone (hidden size 768, 12 heads, patch size 2), class conditioning over the 10 CIFAR-10 classes with label dropout 0.1, and EMA decay 0.9999, and both are trained for 400k iterations. We compare (i) a single tied block applied $K = 20$ times with the internal clock, and (ii) the identical architecture with the clock removed from the AdaLN modulation, so that the block acts identically at every step. Samples are generated in one step ($t = 1$, $r = 0$) without guidance, and FID is computed against CIFAR-10 training-set statistics.

Model	Blocks	K	Params	Clock	LR (schedule)	Batch	Steps	FID-50k
Tied + clock	1	20	14.4M	✓	10^{-4} (cosine)	64	400k	**21.27**
Tied, no clock	1	20	14.4M	✗	10^{-4} (cosine)	64	400k	**45.45**

Table 4: **Internal-clock ablation on CIFAR-10**, trained from scratch. The two models share architecture, parameters, compute, and optimization settings, and differ only in the internal clock. Removing it more than doubles FID-50k ($21.27 \rightarrow 45.45$), showing that the clock is essential to the tied block.

Results. Removing the clock from the tied block more than doubles FID-50k ($21.27 \rightarrow 45.45$; Table 4). Weight tying therefore works only when each application is told where it is along the computational trajectory.

C.5 DRIFTING MODELS AS A NEGATIVE CONTROL

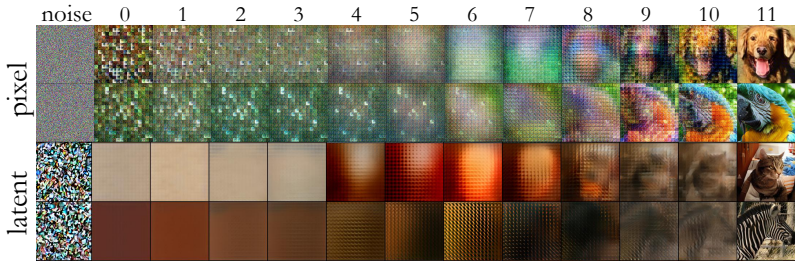


Figure 14: Applying $\mathbf{W}_{\text{out}}$ to the residual states of pixel- and latent-space drifting `DiT-B/2` models does not reveal the depth-as-time structure observed in task-conditioned models.

Drifting models (Deng et al., 2026) constrain the endpoint distribution without prescribing a time-indexed transport task. Although they can still refine their predictions over depth, their intermediate residual states do not form the same readable trajectory under the model's own $\mathbf{W}_{\text{out}}$ readout; recognizable structure emerges primarily in later blocks (Fig. 14).

Distillation recipe. We distill the latent-space drifting `DiT-B/2` (12 blocks, hidden size 768, 12 heads, 16 register tokens) into a single tied block with $K = 12$ and the internal clock. The recipe follows the MeanFlow one with two differences. First, because the norm of the teacher's residual stream grows by roughly $141.5\times$ across depth (as opposed to MeanFlow's $22.9\times$), we match ℓ_2-normalized states, supervising their direction. Second, because drifting models have no transport conditioning, the student receives only the class label and the interval (τ_i, τ_{i+1}), and its output head predicts the image directly.

Model	Params	Compression	FID-50k	Degradation
Drifting `B/2` (teacher)	132.7M	–	1.75	–
one tied block, $K = 12$	15.8M	$8.4\times$	47.09	$\approx 27\times$
MeanFlow `SiT-B/2` (teacher)	131M	–	6.17	–
one tied block, $K = 12$	16.0M	$8.2\times$	15.61	$2.5\times$

Table 5: **Compressing a drifting model versus a MeanFlow model** into a single time-conditioned block (pre-FD, ImageNet 256×256). Both teachers are compressed by a similar factor, yet the drifting student degrades far more than the MeanFlow student, despite distilling from a stronger teacher.

Results. As Table 5 shows, this procedure does not produce an effective compressed drifting model: at a similar compression ratio, the drifting student reaches FID 47.09, about three times worse than the MeanFlow student (15.61).

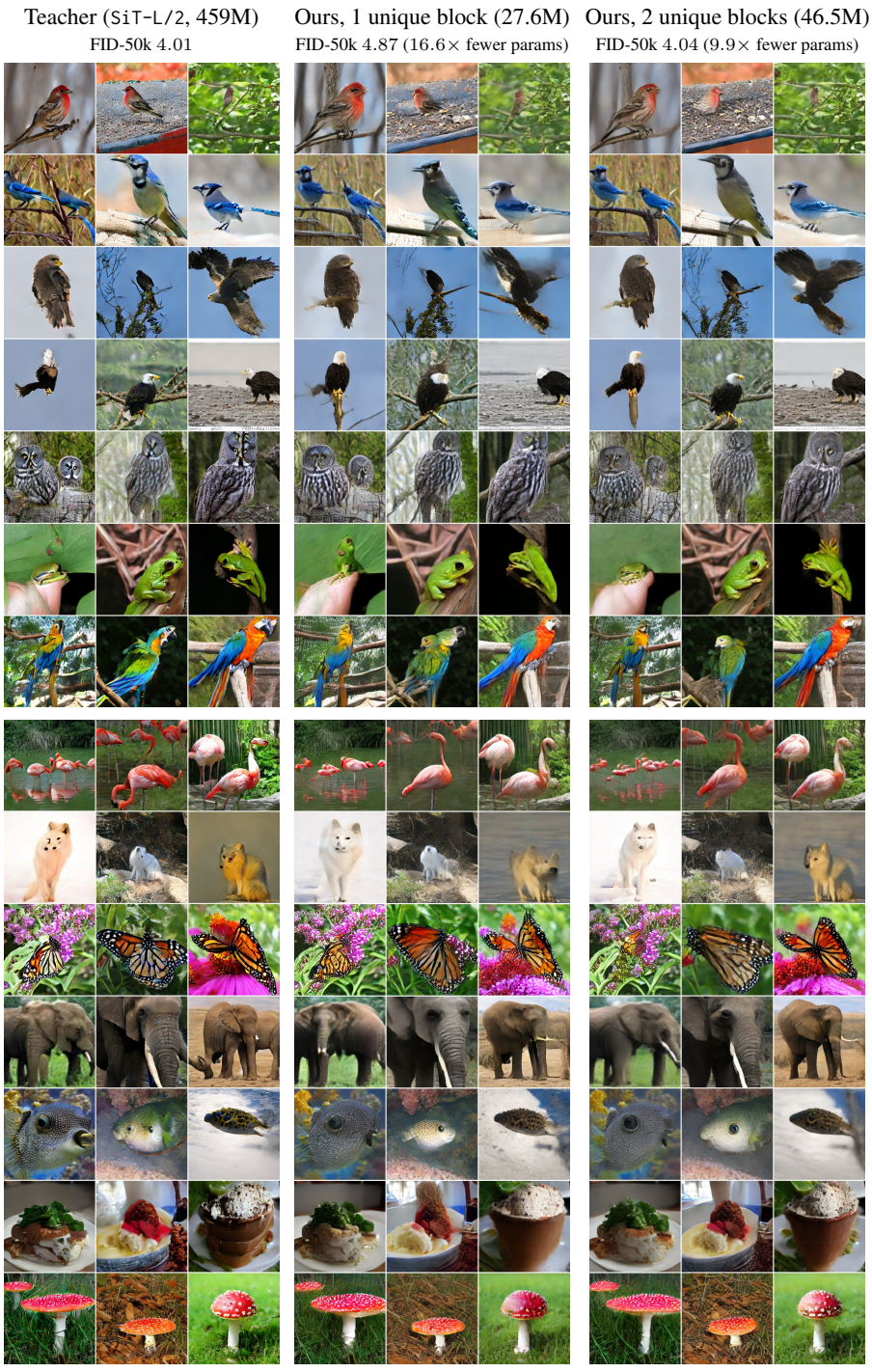


Figure 15: **Samples from the MeanFlow SiT-L/2 teacher and its compressed students** on ImageNet 256×256. Each row within a panel is one class, and corresponding positions across the three columns use the same initial noise and class label. Students are fixed-K models with $K = 24$ after FD-loss post-training (Table 1). Despite using 16.6× and 9.9× fewer parameters, the students largely preserve the teacher's composition, pose, and color for each sample, with differences mostly in fine detail.

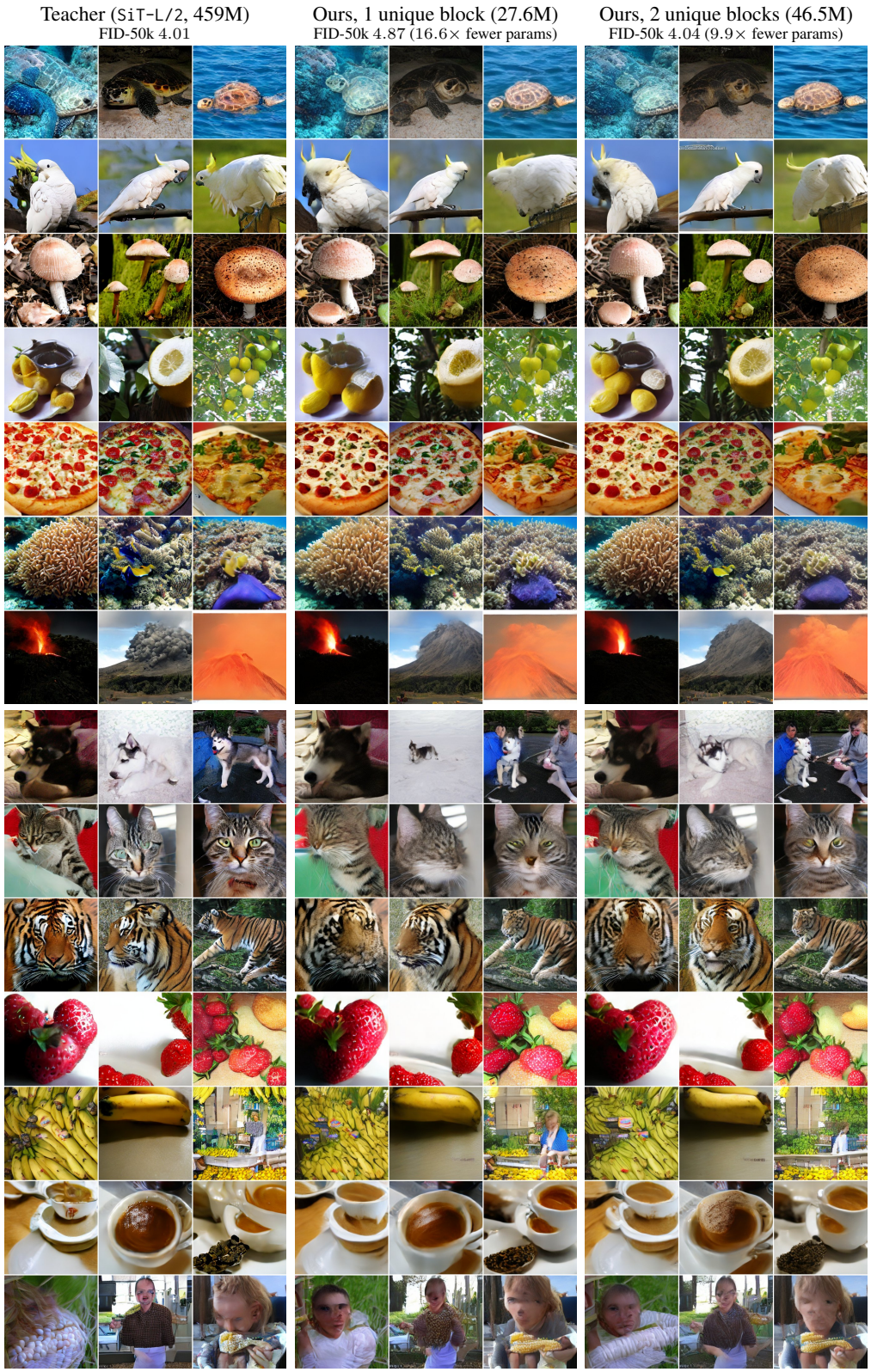


Figure 16: **Additional samples from the MeanFlow SiT-L/2 teacher and its compressed students**, arranged as in Figure 15: each row within a panel is one class, and corresponding positions across the three columns use the same initial noise and class label.

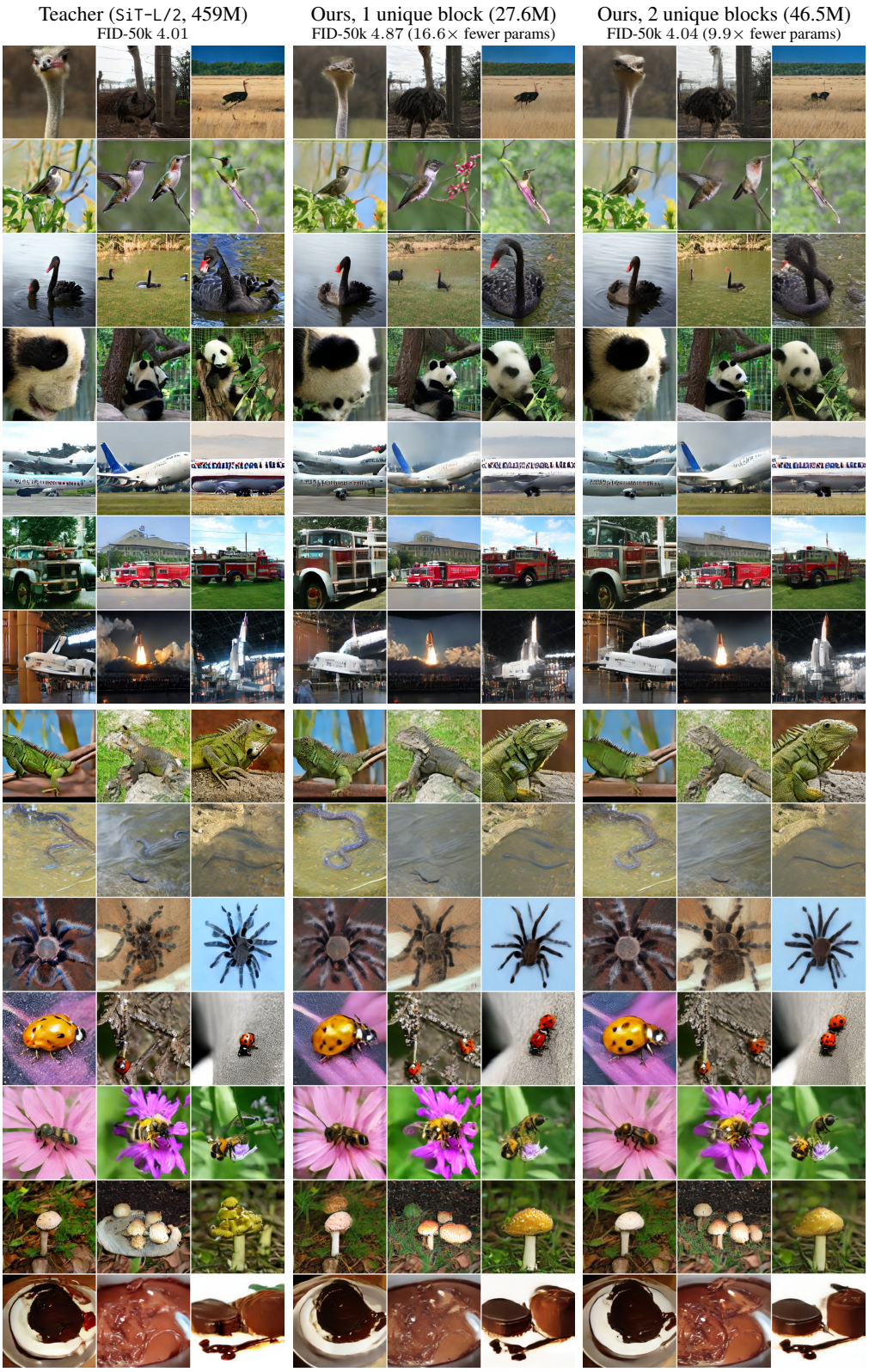


Figure 17: **Additional samples from the MeanFlow SiT-L/2 teacher and its compressed students**, arranged as in Figure 15: each row within a panel is one class, and corresponding positions across the three columns use the same initial noise and class label.

D Extra Background on One-Step Generators

This appendix reviews the training objectives of the one-step generators analyzed in the main paper, following the taxonomy of Section 2. We first cover distillation, where a one-step student is trained against a multi-step teacher (App. D.1). We then cover flow maps trained with self-consistency objectives, both boundary-anchored and interval-conditioned (App. D.2), and step-conditioned (App. D.3). Throughout, we use the convention of Section 3.1: $\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\boldsymbol{\epsilon}$, so that $t = 0$ corresponds to data and $t = 1$ to noise, with instantaneous velocity $v(\mathbf{x}_t, t)$ and flow map $\Phi_{t\to r}(\mathbf{x}_t) \coloneqq \mathbf{x}_t - \int_r^t v(\mathbf{x}_\tau, \tau)\, d\tau$.

D.1 Distillation

Distillation methods train a one-step generator $G_\theta(z)$ to reproduce the output distribution of a multi-step teacher. A representative example is distribution matching distillation (DMD) (Yin et al., 2024), commonly expressed as a reverse KL divergence between the noised student and teacher distributions, $\mathcal{L}_{\mathrm{DMD}}(\theta) \coloneqq \mathbb{E}_t\left[\omega(t)\, \mathbb{E}_{\mathbf{x}_t^\theta \sim p_t^\theta}\left[\log p_t^\theta(\mathbf{x}_t^\theta) - \log p_t(\mathbf{x}_t^\theta)\right]\right]$, where $\omega(t)$ is a time-dependent weighting function. The distributions p_t and p_t^θ use the same Gaussian perturbation kernel $p_t(\mathbf{x}_t \mid \mathbf{x}_0)$ but differ in their clean-data distributions: p_{data} for real samples and p_0^θ for samples produced by the student. The optimum is attained when $p_0^{\theta^*} = p_{\mathrm{data}}$. Because neither $\log p_t$ nor $\log p_t^\theta$ is available in closed form, DMD estimates the generator gradient from the difference between real and fake score functions,

$$\nabla_\theta \mathcal{L}_{\mathrm{DMD}} = \mathbb{E}_{z,\,t,\,\epsilon}\left[\omega(t)\left(s_{\mathrm{fake}}(\mathbf{x}_t^\theta, t) - s_{\mathrm{real}}(\mathbf{x}_t^\theta, t)\right)^\top \frac{\partial G_\theta(z)}{\partial \theta}\right], \tag{8}$$

where $\mathbf{x}_t^\theta = \alpha_t G_\theta(z) + \sigma_t \boldsymbol{\epsilon}$. Here, s_{real} and s_{fake} estimate the scores of p_t and p_t^θ, respectively, and s_{fake} is detached when differentiating the generator. The score difference $s_{\mathrm{fake}} - s_{\mathrm{real}}$ is contracted with the student Jacobian $\partial G_\theta / \partial \theta$, supervising the student across noise levels t.

The distilled models we analyze use related objectives. FLUX-schnell is trained with latent adversarial diffusion distillation (LADD) (Sauer et al., 2024a), whose discriminator operates on noised samples at varying diffusion times, and SANA-Sprint (Chen et al., 2025) combines a continuous-time consistency objective with LADD. Although these methods match distributions rather than explicitly reproducing the teacher's sampling trajectory, their learning signal remains indexed by diffusion time. This is one reason to expect their students to organize computation along a denoising-like trajectory across depth, as we observe in Sections 3 and 4.1.

D.2 Boundary-anchored and interval-conditioned flow maps

Consistency models (Song et al., 2023) learn a flow map whose endpoint is fixed to the clean sample, $f_\theta(\mathbf{x}_t, t) \approx \Phi_{t\to 0}(\mathbf{x}_t)$. Subsequent work lifts this restriction and learns transport between arbitrary pairs of times t and r. Consistency trajectory models (CTM) (Kim et al., 2024) and MeanFlow (MF) (Geng et al., 2025) are two instances of this more general formulation.

Parameterization. CTM parameterizes the map from t to r with a displacement g_θ, whereas MeanFlow parameterizes the same transport with an interval-averaged velocity $u_\theta(\mathbf{x}_t, t, r) \approx \frac{1}{t-r}\int_r^t v(\mathbf{x}_\tau, \tau)\, d\tau$. Both approximate the flow map:

$$\Phi_{t\to r}(\mathbf{x}_t) \approx \underbrace{\tfrac{r}{t}\,\mathbf{x}_t + \tfrac{t-r}{t}\, g_\theta(\mathbf{x}_t, t, r)}_{\mathrm{CTM}} = \underbrace{\mathbf{x}_t - (t-r)\, u_\theta(\mathbf{x}_t, t, r)}_{\mathrm{MF}}, \tag{9}$$

mapping the current state $\mathbf{x}_t$ to the endpoint $\mathbf{x}_r$.[4]

Training. Both methods construct their targets largely from the model's own predictions. Writing $G_\theta(\mathbf{x}_t, t, r)$ for CTM's estimate of $\Phi_{t\to r}(\mathbf{x}_t)$, CTM enforces self-consistency by requiring a direct map from t to r to agree with one that passes through an intermediate time $s \in [r, t]$,

[4] We refer the reader to Chapter 11 of Lai et al. (2025) for a detailed relationship between the two parameterizations.

$\mathcal{L}_{\mathrm{CTM}}(\theta) = \mathbb{E}_{t,s,r}\,\mathbb{E}_{\mathbf{x}_t\sim p_t}\left[\left\|G_\theta(\mathbf{x}_t,t,r) - G_{\theta^-}\left(\hat{\Phi}_{t\to s}(\mathbf{x}_t),s,r\right)\right\|_2^2\right]$, where $\hat{\Phi}_{t\to s}$ is obtained by numerically solving a pretrained teacher's ODE from t to s, and θ^- denotes detached parameters. MeanFlow instead trains the interval-averaged velocity using the instantaneous velocity together with a correction involving the total derivative of the interval average, $\mathcal{L}_{\mathrm{MF}}(\theta) = \mathbb{E}_{t,r}\,\mathbb{E}_{\mathbf{x}_t\sim p_t}\left[\left\|u_\theta(\mathbf{x}_t,t,r) - \mathrm{sg}\big(v(\mathbf{x}_t,t) - (t-r)\frac{d}{dt}u_\theta(\mathbf{x}_t,t,r)\big)\right\|_2^2\right]$, where $\mathrm{sg}(\cdot)$ denotes the stop-gradient operator.

Clean versus interval endpoints. The distinction between clean-endpoint and interval-endpoint prediction is central to the depthwise behavior we observe. When $r = 0$, the model transports $\mathbf{x}_t$ directly to the clean sample $\mathbf{x}_0$. In a multi-step flow map, however, r may be greater than zero, so the target $\mathbf{x}_r$ remains a noisy intermediate state. MeanFlow is therefore trained to predict an average velocity anchored to the interval endpoint $\mathbf{x}_r$, rather than always to the clean image $\mathbf{x}_0$. As we show in Section 4.2, this difference in the training objective changes the structure of the trajectory across depth, producing the denoise-then-renoise pattern.

D.3 Step-conditioned flow maps: Shortcut models

Shortcut models (Frans et al., 2025) condition the network not only on the current time t but also on a desired step size d. The model learns a *shortcut* $s_\theta(\mathbf{x}_t,t,d)$ that transports $\mathbf{x}_t$ to the state a distance d closer to the data,[5] $\mathbf{x}'_{t-d} = \mathbf{x}_t - d\,s_\theta(\mathbf{x}_t,t,d)$, so that $r = t-d$ in the notation of Section 2. As $d\to 0$, the shortcut reduces to the instantaneous velocity.

Rather than regressing onto a teacher-defined target, Shortcut models exploit a self-consistency relation: one step of size $2d$ should agree with two consecutive steps of size d, $s_\theta(\mathbf{x}_t,t,2d) = \frac{1}{2}s_\theta(\mathbf{x}_t,t,d) + \frac{1}{2}s_\theta(\mathbf{x}'_{t-d},t-d,d)$. This yields a joint objective that combines flow matching at $d=0$ with self-consistency at $d>0$,

$$\mathcal{L}_S(\theta) = \mathbb{E}_{\mathbf{x}_0\sim p_{\mathrm{data}},\,\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\,(t,d)}\left[\|s_\theta(\mathbf{x}_t,t,0) - (\boldsymbol{\epsilon}-\mathbf{x}_0)\|_2^2 + \|s_\theta(\mathbf{x}_t,t,2d) - s_{\mathrm{target}}\|_2^2\right], \tag{10}$$

where $s_{\mathrm{target}} = \mathrm{sg}\big(\frac{1}{2}s_\theta(\mathbf{x}_t,t,d) + \frac{1}{2}s_\theta(\mathbf{x}'_{t-d},t-d,d)\big)$ is constructed from the model's own predictions. The flow-matching term anchors the model to the instantaneous velocity $\boldsymbol{\epsilon}-\mathbf{x}_0$, while the self-consistency term propagates this capability from short steps to progressively larger ones, including the one-step setting.

E Extra Analyses on Flow Maps

E.1 Shortcut models

Setup.

Shortcut models (Frans et al., 2025) differ from MeanFlow in their task conditioning: instead of an endpoint time r, the network receives the current noise level t and a step size d, so that $\boldsymbol{\tau}_{\mathrm{task}} = (t,d)$ and the endpoint is $r = t-d$ (Appendix D.3). We compute layerwise predictions (Eq. (2)) for a DiT-B/2 Shortcut model s_θ trained on ImageNet (Russakovsky et al., 2015), using the two-step schedule $1\to 0.5\to 0$, which corresponds to the two network evaluations $s_\theta(\mathbf{z}_1,1,0.5)$ and $s_\theta(\mathbf{z}_{0.5},0.5,0.5)$ (Alg. E.1). From the shortcut predicted at block i, s_i, we decode the endpoint prediction with the model's own update rule, $\tilde{\mathbf{x}}^{\,i}_{t-d} = \mathbf{z}_t - d\,s_i$, and the diagnostic clean prediction $\tilde{\mathbf{x}}^{\,i}_0 = \mathbf{z}_t - t\,s_i$. The first evaluation therefore decodes blockwise predictions of the noisy state $\mathbf{z}_{0.5}$, and the second decodes blockwise predictions of $\mathbf{z}_0$.

Algorithm E.1 Shortcut Depth Probe

```
z ← z_1
for (t, d) ∈ {(1, 0.5), (0.5, 0.5)} do
  {h_i}_{i=1}^L ← s_θ(z; t, d)
  for i = 1, ..., L do
    s_i ← W_out(h_i; t, d)
    x̃^i_{t−d} ← z − d s_i
    x̃^i_0 ← z − t s_i
  end for
  z ← z − d s_L
end for
```

[5] We state Shortcut models in our time convention, in which $t=0$ is data; the original formulation uses the reverse convention.

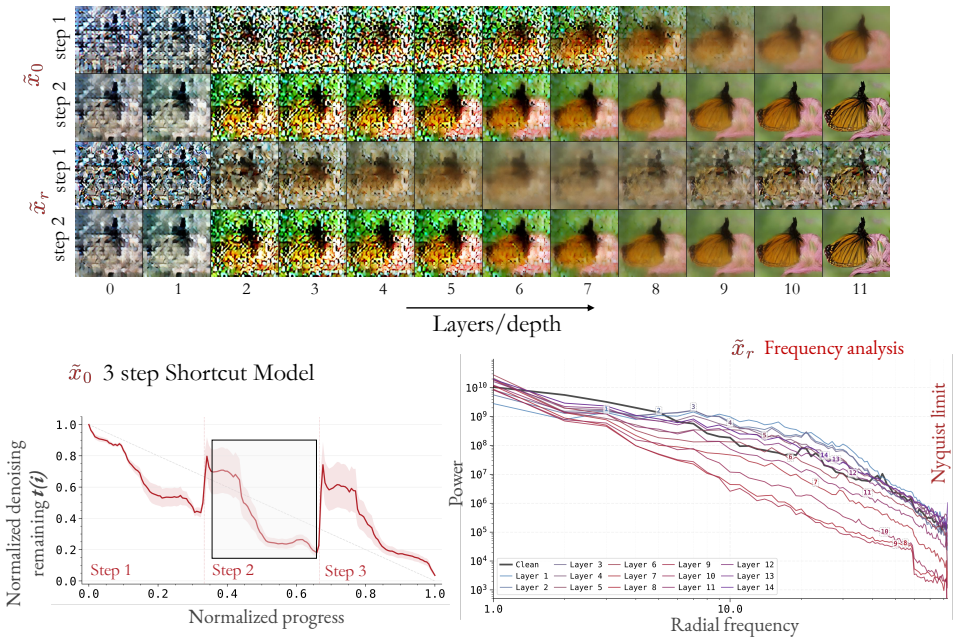


Figure 18: **Depth as time in a Shortcut model.** *Top:* endpoint predictions $\tilde{\mathbf{x}}^i_{t-d}$ across depth during a two-step rollout. Within the first step, whose endpoint is noisy, predictions first reveal cleaner structure and then return toward the prescribed endpoint. *Bottom left:* normalized distance $\rho(i)$ computed from the diagnostic clean predictions $\tilde{\mathbf{x}}^i_0$. *Bottom right:* radially averaged power spectra of the endpoint predictions in the first step. Intermediate blocks move toward the clean-image spectrum before later blocks return toward the noisy endpoint spectrum.

Observations. The Shortcut model shows the same denoise-then-renoise pattern as MeanFlow (Section 4.2). In the first step, whose endpoint $\mathbf{x}_{0.5}$ is noisy, the layerwise predictions first reveal cleaner structure and then return toward the endpoint in the final blocks (Fig. 18, top), and $\rho(i)$ correspondingly decreases and then increases within the step (Fig. 18, bottom left). Depth as time thus extends to step-conditioned flow maps.

Frequency-domain signature. We also compute the radially averaged power spectrum of each endpoint prediction in the first step (Field, 1987; Simoncelli and Olshausen, 2001): for block i, $P_i(\omega) = |\mathcal{F}[\tilde{\mathbf{x}}^i_{t-d}](\omega)|^2$, averaged over frequencies of equal magnitude $\kappa = \|\omega\|_2$ to give $\bar{P}_i(\kappa)$. Intermediate blocks shift power toward the clean-image spectrum, and later blocks shift it back toward the noisier endpoint spectrum (Fig. 18, bottom right), confirming the pattern in frequency space.