DINO FORCING FLOW MODELS: DO NOT DENOISE WHAT YOU CAN PREDICT

Anonymous authors
Paper under double-blind review

ABSTRACT

Co-denoising pretrained representations such as DINO can substantially improve the training speed and quality of flow matching models, but it introduces a second denoising trajectory and requires carefully designed schedules. We propose a simpler alternative: predict the pretrained representation directly, then condition the model on its own prediction. This removes the need for a second ODE and any representation-specific denoising schedules, while retaining the benefits of representation guidance. Our approach converges substantially faster and achieves better generation quality as measured by FID score. On ImageNet, it outperforms the state of the art in latent space at 2× fewer epochs than prior methods; in pixel space, it improves over comparable prior methods by more than 20%. These results support a simple principle: do not denoise what you can predict. We will release all code and models.

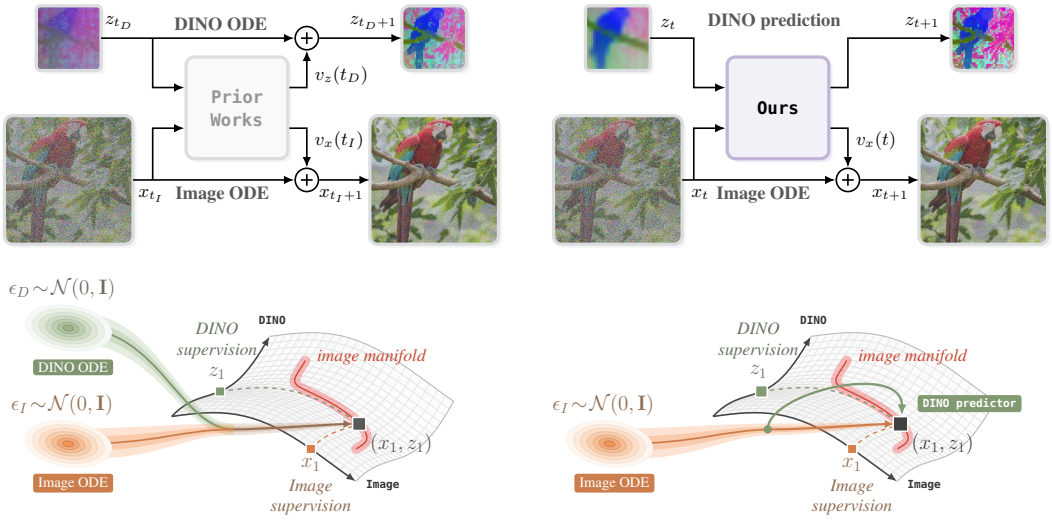

Figure 1: **Left:** Prior co-denoising methods generate both the image and the pretrained representation from noise by jointly solving their individual ODEs through iterative modification during sampling. The model predicts the velocity for both modalities at different time schedules (*top*) which leads to 2 trajectories starting from independent noise samples and arriving at the same pair (*bottom*). **Right:** Our proposed **`Dino Forcing`** (**`DF`**) method keeps the ODE only for the image generation since the representation is deterministically tied to the image. The model predicts the target representation in addition to the image velocity (*top*), providing a self-condition to guide the image ODE (*bottom*).

1 INTRODUCTION

Pretrained representations should help a model generate better images; not strain it with learning one more diffusion trajectory! Yet recent flow matching methods that leverage pretrained representations, such as `DINO`, typically diffuse and denoise the representation alongside the image. In fact, a trend started by (Bao et al., 2023; Xu et al., 2023) and recently popularized by (Kouzelis et al., 2026b;

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.

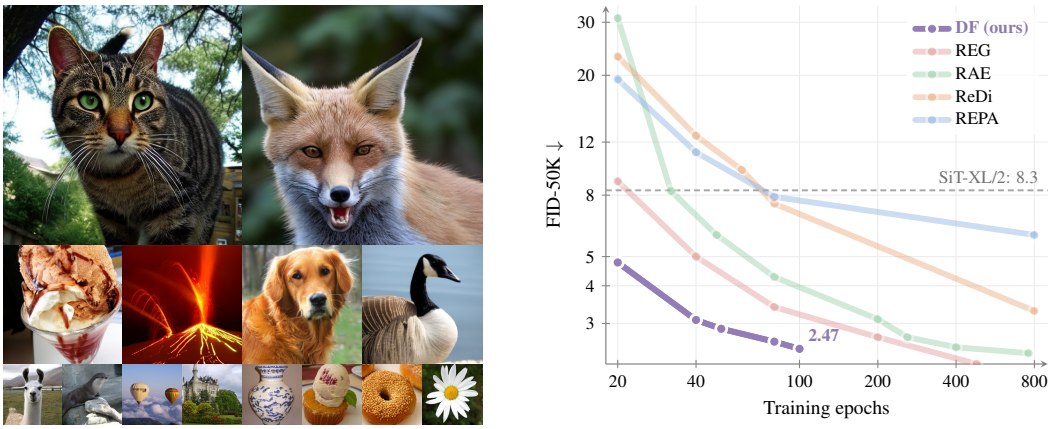

Figure 2: **Left:** Qualitative samples. More examples are given in Figures 8 to 13 **Right:** Convergence speed-up (FID-50K vs. training epochs).

Baade et al., 2026) is to jointly denoise images and DINO embeddings, followed by (Wu et al., 2026; Pan et al., 2026; Petsangourakis et al., 2025; Lin et al., 2026) that extend this with CLS token, dual-stream or auxiliary networks. This joint co-denoising speeds-up training compared to the standard denoising as the model can leverage the additional information of the representation.

However, in this work, we claim that this co-denoising comes with two paradoxes: First, the extra information introduced to guide the image-generation problem is now completely hidden in the noise such that the model has to untangle the informative bits of that modality. Second, the learning problem is made artificially harder than it should be[1] because it consists of matching pairs of independent noise samples (ϵ_I, ϵ_D from Fig. 1, bottom left) to a coupled pair (image x, DINO $z = \phi(x)$) that does not require the extra noise since the link between the image and its representation is deterministic.

In this work, we remove these paradoxes with one simple idea: pretrained representations should constrain generation, not become another generation target. The pretrained representation is not an independent quantity that needs its own generative trajectory; it is a property of the image that the model can progressively infer as the image emerges. Our key idea is therefore simple: rather than denoising a second representation, we ask the model to *predict* the representation it needs (Fig. 1 top right). This keeps the search entirely in the image space, recovering the simple one-to-one mapping problem (Fig. 1 bottom right) while benefiting from the guidance of the pretrained representation.

This principle leads directly to our proposed DINO Forcing (DF), a method with two ouputs, one from image ODE and one for predicting DINO features that are used as self-conditioning. During training, a first look-ahead pass predicts the DINO representation associated with the current noisy image; a second pass conditions on this prediction to learn the image velocity. The predicted representation is aligned with a frozen DINO encoder (Fig. 3, right). At inference, the same prediction is simply rolled forward from one solver step to the next (Fig. 4). Thus, the pretrained representation is produced by the generator itself and evolves together with the image.

In latent space, DF reaches an FID of 2.47 after only 100 epochs, surpassing prior methods trained for substantially longer (Fig. 2, right). At the same training budget, our B model outperforms XL baseline models, showing that the gain does not come from scale. In pixel space, DF improves over Latent Forcing (Baade et al., 2026) from 7.20 to 5.53 FID at comparable model size and training budget. These results support our claim that explicitly co-denoising is unnecessary: predicting the pretrained representation is not only simpler, but leads to substantially faster convergence.

Our contrbutions are: (1) we show that directly predicting DINO representations is sufficiently informative to guide generation, without needing to explicitly denoise it. (2) we propose DINO Forcing: a schedule-free self-conditioning mechanism that predicts, rather than denoises, the pretrained representation used to guide generation. (3) DF achieves sets the new state of the art in both latent and pixel diffusion, while reaching strong performance in substantially fewer training epochs.

[1] It is still easier than not using the additional representation.

2 Related Work

Representation alignment in image generation. Diffusion (Sohl-Dickstein et al., 2015; Ho et al., 2020) and flow matching (Lipman et al., 2023) models are trained with denoising objectives that favour fine-grained, reconstruction-oriented representations over high-level semantic ones (Yu et al., 2025), which slows convergence: state-of-the-art generators typically need over 500 epochs (`DiT`: Peebles & Xie, 2023; `SiT`: Ma et al., 2024; `JiT`: Li & He, 2026; `fv-loss`: Degeorge et al., 2026; `TREAD`: Krause et al., 2025). Aligning internal features with external pretrained representations accelerates training (`REPA`: Yu et al., 2025; `iREPA`: Singh et al., 2026a; `HASTE`: Wang et al., 2025b; `REED`: Wang et al., 2025a; `DDT`: Wang et al., 2026; `PixelDiT`: Yu et al., 2026a; `PixelREPA`: Shin et al., 2026; `DeCo`: Ma et al., 2026a; `PixelGen`: Ma et al., 2026b). But at sampling time the model runs unguided: the signal lacks when it is most needed. A complementary line performs flow matching directly in a pre-aligned latent space (`RAE`: Zheng et al., 2026; `DINO-SAE`: Chang et al., 2026), so the representation is present throughout sampling. But this requires training a new decoder and architectural changes. A more comprehensive comparison is given in Appendix D.

Co-denoising for image generation. Jointly denoising multiple modalities (Bao et al., 2023; Xu et al., 2023) offers an appealing alternative: it implicitly aligns pretrained representations while preserving them throughout generation, including at inference. This accelerates training convergence, both in a latent (`ReDi`: Kouzelis et al., 2026b; `CoReDi`: Kouzelis et al., 2026a; `REG`: Wu et al., 2026; `SFD`: Pan et al., 2026; `REGLUE`: Petsangourakis et al., 2025) and directly in pixel space (`LatentForcing`: Baade et al., 2026; `V-Co`: Lin et al., 2026; `CoReDi`: Kouzelis et al., 2026a). The cost is a second noise scheduler that must be hand-crafted: `SFD` (Pan et al., 2026) and `LatentForcing` (Baade et al., 2026) both find that denoising the representation to a low noise level *before* starting the image yields the fastest convergence, a schedule that likely needs re-tuning for different modalities. A more direct approach is to condition the generator on pretrained representations rather than denoise them from scratch (`RCG`: Li et al., 2024; `PixelDiT2`: Yu et al., 2026b). Inspired by self-conditioning (Jabri et al., 2023; Chen et al., 2023), we condition the model on its *own predicted representation* at each denoising step. Unlike prior self-conditioning work, we align these predictions with an external pretrained feature space, i.e. `DINO`. The representation is then available throughout generation with no handcrafted schedule, while still training faster.

3 DINO-Forcing

Here, we first give an overview of `DF`, then detail its two-pass training procedure (Section 3.1), and how image and representation tokens are fused, and finally its sampling (Section 3.2).

Overview `DINO-Forcing` (`DF`) generates images by relying on a simple idea: rather than denoising a separate pretrained representation in parallel, it *predicts* the pretrained representation it needs (i.e. `DINO`) and conditions itself on that prediction throughout generation.

Concretely, at each training step, `DF` produces a *look-ahead* estimate of the pretrained representation, $\hat{z}$, associated with the image, x, being denoised (see Fig. 3; Pass-1). Then, it uses that estimate as a conditioning signal for a forward-backward update pass (see Fig. 3; Pass-2). At sampling time, these `DINO` representations are predicted and are rolled out across denoising steps, providing a continuously updated, model-generated guide (see Fig. 4).

3.1 Self-Conditioning on Predicted Representations

The core idea of our proposed `DF` is a two-pass training procedure, illustrated in Fig. 3 and detailed as follows. For the remainder of this section, for simplicity, we omit referring to additional conditions, such as class or text; however, in our experiments we do include these conditions.

Pass 1: Representation look-ahead prediction. We perform a forward pass with x_t as the image input and a zeros tensor, $\mathbf{0}$, as placeholder for the representation conditioning. This produces a preliminary representation prediction, $\hat{z}^{(\Phi)}$, which serves as a look-ahead estimate of the content in the current denoising trajectory (similar to Jabri et al. (2023); Chen et al. (2023); Dufour et al. (2024); left part of Fig. 3). This pass runs without gradients: it provides the conditioning for Pass 2.

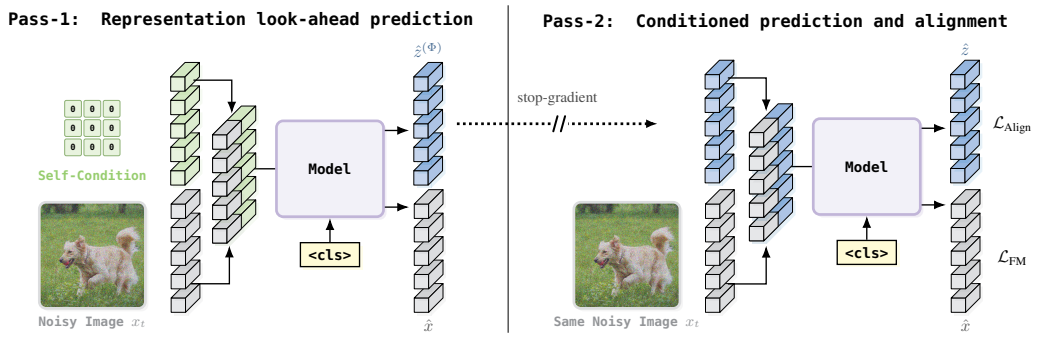

Figure 3: **Overview of our proposed training framework**. A first forward pass produces a representation look-ahead (no gradient), which is then used to condition a second pass that optimizes both the flow matching and representation alignment objectives.

Pass 2: Conditioned prediction and alignment. The look-ahead prediction $\hat{z}^{(\Phi)}$ is then used as the conditioning input in a second forward pass. This pass produces two outputs: (i) the flow matching velocity prediction $v_\theta(x_t|\hat{z}^{(\Phi)}, t)$ for image generation, and (ii) a refined representation prediction $\hat{z}$ that is supervised against an external pretrained feature space, $\phi(x)$ (contrary to prior methods Jabri et al. (2023); Chen et al. (2023); Dufour et al. (2024) who leave this added condition unsupervised). As such, the final training objective is formulated as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{FM}} + \lambda \cdot \mathcal{L}_{\text{Align}}, \tag{1}$$

with λ a scalar balancing the two objectives: $\mathcal{L}_{\text{FM}} = \|v_\theta - (x - \epsilon)\|^2$ is the standard flow matching velocity loss (Lipman et al., 2023), and $\mathcal{L}_{\text{FR}}$ is the loss between the predicted and frozen representation. In this work, we follow `REPA` formulation (Yu et al., 2025) and define $\mathcal{L}_{\text{Align}}$ as a cosine similarity loss between $\hat{z}$ and target representations $\phi(x)$ extracted from the clean image x. This is given as follows:

$$\mathcal{L}_{\text{Align}} = -\left\langle \frac{\hat{z}}{\|\hat{z}\|}, \frac{\phi(x)}{\|\phi(x)\|} \right\rangle \tag{2}$$

In this work, $\phi(x)$ denotes `DINO` features, following (Kouzelis et al., 2026b; Baade et al., 2026). Note that the formulation is not tied to this choice: any other comparative loss could be used instead.

With this two-pass design, the model learns two things jointly: generating images and predicting accurate representations. These predictions are fed back as conditioning in the same training step.

Fusing Image and Frozen Representations. `DF` takes as input merged image and pretrained representation (i.e. `DINO`) tokens. For merging, we concatenate them along the channel dimension (see Fig. 3), keeping the token count identical to the standard image-only model, which makes our work compatible with existing flow models. More details in Appendix B, and a discussion of other fusion strategies in Lin et al. (2026).

3.2 Sampling

At inference, we replicate the structure of training through a representation roll-out across denoising steps, illustrated in Fig. 4 (similar to Jabri et al. (2023); Dufour et al. (2024)).

We initialize the image input with pure noise $x_0 \sim \mathcal{N}(0, I)$ and set the starting representation to a zero tensor $\mathbf{0}$, mirroring Pass 1 at training time (see Section 3.1). At each denoising step t, the model produces both a velocity estimate, used to update x_t via the ODE solver, and a representation prediction $\hat{z}$. This prediction is then passed as the conditioning input for the *next* denoising step, replacing the zeros placeholder. The roll-out continues for all steps of the solver.

The result is a sampling procedure in which the representation conditioning signal grows progressively more coherent as the image takes shape (see Fig. 5 and Fig. 14), providing structured guidance precisely when it is most useful, without any modification to the standard denoising schedule.

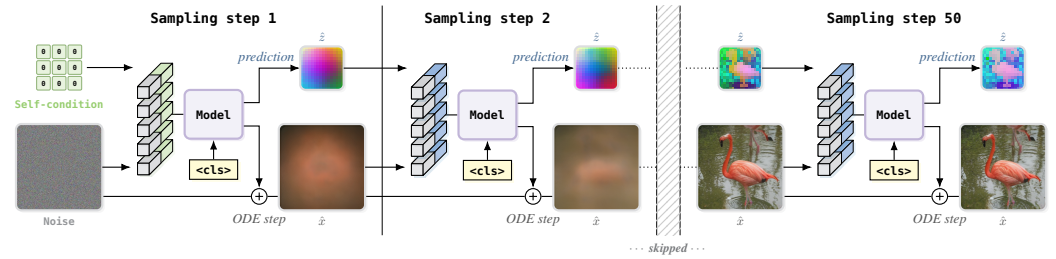

Figure 4: **Overview of our sampling framework.** At each step the model outputs a *ODE step*, and a *prediction* $\hat{z}$ used as the self-condition at the next step. Note that $\hat{z}$ is not a denoised feature map but the model's prediction of the DINO features of the denoised image $\hat{x}$.

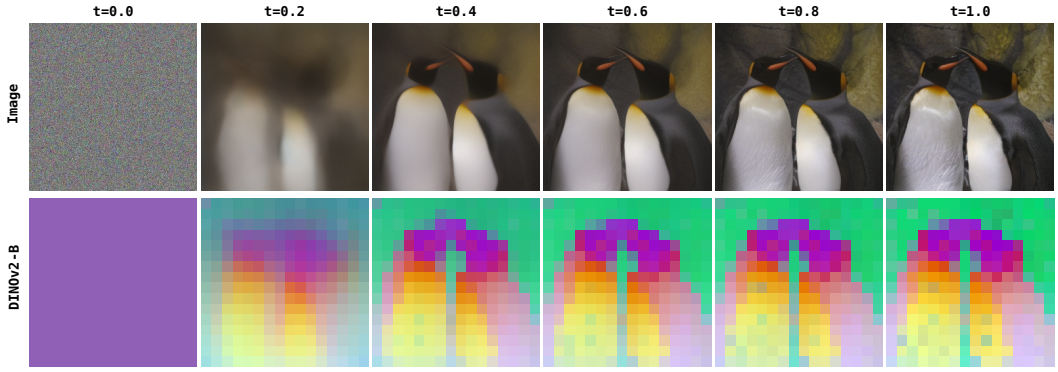

Figure 5: Evolution of the image and the PCA of the predicted `DINO` features throughout sampling.

4 Prediction vs co-denoising

As motivated in Section 2, most recent image generation approaches either remove pretrained representation information at inference time, or require extra decoders and hand-crafted denoising schedules to preserve it.

We focus on co-denoising, as it keeps an estimate of the pretrained representation during the denoising process, and try to directly address the limitations of these approaches. Co-denoising methods (Bao et al., 2023; Kouzelis et al., 2026b; Baade et al., 2026) come with notable performance gains, mainly due to sharing one network between two tasks: denoising the image x and denoising an additional pretrained representation $\phi(x)$, which can be text as in (Bao et al., 2023) or `DINOv2` as in (Kouzelis et al., 2026b; Baade et al., 2026). Even though this results in faster convergence than image-only denoising baselines, it requires an additional ODE trajectory for the pretrained representations (and thereby handcrafted schedules).

Specifically, with independent noises ϵ_I, ϵ_D and two time variables, the co-denoising model fits

$$v_\theta\big(x_{t_I},\, \phi(x)_{t_D},\, t_I,\, t_D\big), \qquad (t_I, t_D) \in [0,1]^2 \quad ,$$

so it must be accurate over the whole square of noise-level pairs (Fig. 1, bottom left).

Instead, we claim that this additional trajectory can be omitted. Since ϕ is deterministic, the pair $(x, \phi(x))$ never fills the product space, but it lies on

$$\{(x, \phi(x)) : x \in \mathcal{X}\} \subset \mathbb{R}^{d_I} \times \mathbb{R}^{d_D} \quad ,$$

and the pretrained representation coordinate adds no degrees of freedom to the distribution being modeled: given a clean image there is no distribution over $\phi(x)$ to sample from. Given a noisy x_t, the best estimate of the pretrained representation is $\hat{z} = \mathbb{E}[\phi(x) \mid x_t]$, which depends on x_t only with no noise level of its own. A forward pass of the network is enough to predict it, and needs no schedule. As such, in `DINO Forcing` asks the model to simply *predict* the pretrained representation it needs and condition itself on that prediction throughout generation (Fig. 4).

This results in us keeping the shared network and its two tasks, but it removes the quadratic search of two ODE trajectories (the time domain collapses from $[0, 1]^2$ back to $[0, 1]$) (Fig. 1, right) and the corresponding handcrafted schedules.

5 EXPERIMENTS

We evaluate our method on the ImageNet class-conditional benchmark, in both latent space (Section 5.1) and pixel space (Section 5.2). We then analyse what the representation prediction branch learns by inverting its predictions back to images (Section 5.3), and study generation driven by the predicted features alone (Section 5.4). We ablate the components of our method in Section 5.5. Qualitative samples are shown in Fig. 2 left, with more uncurated samples in Appendix E.

Evaluation Metrics We compare **DF** against latent space state-of-the-art methods using the `ADM` (Dhariwal & Nichol, 2021) evaluation suite and its reference statistics. We generate 50K samples and report Fréchet Inception Distance (**FID**) (Heusel et al., 2017), Inception Score (**IS**) (Salimans et al., 2016), Precision (**P**) and Recall (**R**) (Kynkäänniemi et al., 2019). For pixel space, we report **FID** against the `JiT` reference statistics. Since `JiT` does not release the images needed for the remaining metrics, **IS**, **P** and **R** are computed with the `ADM` suite.

5.1 LATENT SPACE SETTING

Setup In the latent space setting, we follow the implementations of `SiT` (Ma et al., 2024) with the only exception of having extra input and output projections for the pretrained representation features. We train our model in the `sd-vae-ft-ema` (Rombach et al., 2022) latent space. We follow prior works (Yu et al., 2025; Baade et al., 2026; Kouzelis et al., 2026b) and use `DINOv2-B` (Oquab et al., 2024) features. We weight our frozen representation loss (λ in Eq. (1)) with a value of 0.5, similar to prior work (Yu et al., 2025). Consistent with previous works (Ma et al., 2024), we use the Adam (Kingma & Ba, 2014) optimizer without any weight decay and a constant learning rate of $1e{-}4$. In contrast to previous methods, we sample using a simple Euler ODE solver in 50 steps instead of using SDE solvers. We train using the null condition 10% of the time for classifier-free guidance (Ho & Salimans, 2021). For the pretrained representation condition, 10% of the time we do optimization on just the 1st forward pass to make the model learn a better prediction when the conditions are all zeros. We also set the pretrained representation prediction condition to the original `DINOv2` features 5% of the time to make the model more stable when it has already reached the required feature. For XL models, we use a dropout of 0.1 on the middle third of the `SiT` blocks (both for attention and the feedforward layers). Implementation details are given in Table 8.

Results without guidance We first compare **DF** against the baseline `SiT` models in the unguided setting (Table 1). All three sizes improve substantially: `B`, `L` and `XL` all fall below 5 FID within 400K training steps (or 80 epochs), with XL reaching 2.61.

Table 1: Model-size ablation with `DINO` as the frozen representation.

Latent space					*Pixel space*				
Model	**#Params**	**Steps**	**Batch Size**	**FID↓**	**Model**	**#Params**	**Steps**	**Batch Size**	**FID↓**
`SiT-B/2`	130M	400K	256	33.00	`JiT-B/16`	131M	100K	1024	48.61
→ **DF**	138M	400K	256	**4.51**	→ **DF**	139M	100K	1024	**13.37**
`SiT-L/2`	458M	400K	256	18.80	`JiT-L/16`	459M	100K	1024	26.64
→ **DF**	466M	400K	256	**3.04**	→ **DF**	468M	100K	1024	**6.80**
`SiT-XL/2`	675M	400K	256	17.20	`JiT-XL/16`	676M	100K	1024	23.03
→ **DF**	684M	400K	256	**2.61**	→ **DF**	686M	100K	1024	**5.45**

When compared with state-of-the-art works at a similar parameter count (Figure 2), **DF** reaches 2.61 at 80 epochs, better than or on par with methods trained substantially longer: `REPA` at 800 epochs (5.78), `CoReDi` at 400 epochs (3.30), `REG` at 200 epochs (2.70) and `REGLUE` at 200 epochs (2.50). When further trained for 20 more epochs **DF** shows a 2× reduction in training budget over `REGLUE` (see Table 2), the previous best method at a comparable parameter count in the same latent space: 2.47

Table 2: State-of-the-art comparison of latent space models on ImageNet 256^2. **Bold** is best, underline is second best, grayed methods use more than 200 epochs and are not considered for the comparison.

				w/o CFG				w/ CFG			
Model	**DINO use**	**#Params**	**Epochs**	FID↓	IS↑	P↑	R↑	FID↓	IS↑	P↑	R↑
DiT	None	675M	1400	9.62	121.5	0.67	0.67	2.27	278.2	0.83	0.57
SiT	None	675M	1400	8.61	131.7	0.68	0.67	2.06	270.3	0.82	0.59
REPA	Alignment	675M	800	5.78	158.3	0.70	0.68	1.42	305.7	0.80	0.65
RAE-DiT^{DH}-XL	Rep. space	839M	800	1.51	242.9	0.79	0.63	1.13	262.6	0.78	0.67
RAE-DiT-XL	Rep. space	676M	800	1.87	209.7	0.80	0.63	1.41	309.4	0.80	0.63
ReDi	Joint gen.	675M	800	–	–	–	–	1.61	295.1	0.78	0.64
REG	Joint gen.	677M	800	1.80	–	–	–	1.36	299.4	0.77	0.66
DiT + TREAD	None	675M	680	3.93	211.4	0.76	0.60	–	–	–	–
HASTE	Alignment	675M	500	–	–	–	–	1.42	299.5	0.80	0.65
DDT	Alignment	675M	400	6.27	154.7	0.68	0.69	1.26	310.6	0.79	0.65
CoReDi	Joint gen.	675M	400	3.30	176.8	0.74	0.66	1.58	297.2	0.63	0.78
REED	Alignment	675M	200	4.70	–	–	–	1.80	267.5	0.81	0.61
REG	Joint gen.	677M	200	2.70	–	–	–	–	–	–	–
SFD+LDiT/1	Joint gen.	675M	200	2.54	191.7	0.75	0.67	**1.06**	267.0	0.78	0.67
REGLUE	Joint gen.	677M	200	2.50	188.6	0.76	0.65	–	–	–	–
iREPA	Alignment	675M	80	7.13	132.9	0.71	0.66	1.89	**272.4**	0.80	0.60
DF	Forcing	684M	100	**2.47**	**250.1**	**0.84**	**0.68**	2.35	266.1	**0.85**	**0.68**



Figure 6: Reconstructions from inverted DINOv2-B features. **Top** row: originals; **bottom** row: reconstructions.

Table 4: Comparison on ImageNet 512^2. without classifier-free guidance.

Model	FID↓	IS↑	P↑	R↑
SiT-XL/2	20.45	75.1	**0.66**	0.68
REPA	11.05	116.2	0.65	**0.73**
DF (ours)	**3.11**	**274.5**	0.65	0.63

Table 5: DINO inversion quality.

Model	PSNR↑	SSIM↑
DF-SiT-XL	16.86	0.48
DF-JiT-XL	14.33	0.44

after 100 epochs against 2.50 after 200 epochs. We must note that the excellent work, TREAD (Krause et al., 2025), remains competitive without using external pretrained model. Since the two approaches are orthogonal, combining **DF** with TREAD should yield further gains.

Results with guidance Like RAE (Zheng et al., 2026), we found guidance non-trivial to implement for our latent space model. As such, following RAE we also use Autoguidance (Karras et al., 2024) and use an L model's checkpoint at 150K steps to act as the weaker model. This leads to a guided FID score of 2.35 (see Table 2;). It reflects an improvement; however its not as drastic as prior works, which are trained for at least $2\times$ more than our setup. We agree that the guided scores are still a bit underwhelming and more work must be done to improve on it. Interesting ideas like Internal Guidance (Zhou et al., 2026) with intermediate outputs can be an interesting avenue to explore to improve upon the guidance, and we leave this as future work to further improve upon our core idea.

High resolution We next evaluate at 512×512 resolution in the unguided setting (Table 4). **DF** reaches an FID of 3.11 at 80 epochs, against 11.05 for REPA and 20.45 for the SiT-XL/2 baseline. The Inception Score improves from 116.2 to 274.5. Precision is unchanged (0.65), but recall drops from 0.73 to 0.63, indicating the predicted representation condition concentrates the sample distribution: it acts on the trajectory much as guidance would, consistent with the large unguided FID gap.

Table 3: State-of-the-art comparison of pixel-space models on ImageNet 256^2. **Bold** is best, underline is second best, grayed methods use more than 200 epochs and are not considered for the comparison.

				w/o CFG				w/ CFG			
Model	**DINO use**	**#Params**	**Epochs**	FID↓	IS↑	P↑	R↑	FID↓	IS↑	P↑	R↑
JiT-G	None	2B	600	–	–	–	–	1.82	292.6	–	–
JiT-XL + fv-loss	None	676M	600	11.5	118.5	–	–	2.13	290.3	–	–
DeCo-XL	Alignment	682M	600	–	–	–	–	1.69	304.0	0.79	0.63
PixelREPA	Alignment	953M	600	–	–	–	–	1.81	317.2	–	–
PixelDiT2-H	Grounding	1B	480	–	–	–	–	1.48	299.1	–	–
PixelDiT-XL	Alignment	797M	320	–	–	–	–	1.61	292.7	0.78	0.64
V-Co	Joint gen.	1.9B	300	–	–	–	–	1.71	263.3	–	–
JiT-L	None	459M	200	17.19	85.4	0.70	**0.73**	3.06	271.5	**0.85**	0.66
JiT-XL	None	676M	200	12.71	101.7	0.74	**0.73**	2.89	272.1	**0.85**	0.66
LatentForcing-L	Joint gen.	465M	200	7.20	–	–	–	2.48	–	–	–
PixelDiT2-L	Grounding	503M	200	–	–	–	–	2.35	270.1	–	–
PixelGen-XL	Percept.+Align.	676M	160	–	–	–	–	**1.83**	293.6	0.79	0.63
DF-L (ours)	Forcing	468M	200	5.53	163.6	**0.79**	0.72	2.26	309.9	0.84	0.70
DF-XL (ours)	Forcing	686M	200	**4.56**	**179.6**	**0.79**	**0.73**	2.13	**310.0**	0.83	**0.71**

5.2 Pixel Space Setting

Setup For pixel space, we closely follow the implementations of JiT (Li & He, 2026) and we use DINOv2-B as the pretrained representation. We observe that v-loss in pixel-space is $13\times$ lower than in latent space so we lower $\lambda = 0.03$ in Eq. (1). We use Adam optimizer without any weight decay, with a constant learning rate of $2e-4$ after a warmup for 5 epochs. For pixel space experiments, we follow Heun Sampler (Heun et al., 1900) with 50 sampling steps. Similar to the latent space setting, we use the null condition 10% of the training, optimize on the first forward pass 10% of the training, use the original DINOv2-B features for 5% of the training, and we use a dropout rate of 0.1 for our bigger XL models. Additional implementation details are further provided in Table 8.

Results without guidance The improvements observed with SiT carry over to JiT (see Table 1): all model sizes improve with XL reaching an unguided FID of 5.45. We compare **DF** against Latent Forcing (Baade et al., 2026) in Table 3. At 200 epochs, **DF** reaches 5.53 with an L-sized model, against 7.20. We attribute this gap to how the pretrained representation signal is produced: Latent Forcing commits to a single schedule, fixed at training time and shared across the dataset, that dictates when pretrained representation information is revealed relative to pixels. **DF** instead derives the conditioning from the sample being generated, so the pretrained representation signal adapts to each trajectory rather than following a common ordering.

Results with guidance Prior co-denoising works in pixel space, like Latent Forcing (Baade et al., 2026), require a split scheme, applying AutoGuidance (Karras et al., 2024) on pixel timesteps and CFG interval (Kynkäänniemi et al., 2024) on representation timesteps, because the two modalities advance on separate schedules. Having a single trajectory, we need no such split: the default interval guidance setting works well without method-specific tuning (Table 3). At L size we reach 2.26, surpassing Latent Forcing (2.48) as well as PixelDiT-2 (2.35). At XL size we reach 2.13, against 2.89 for JiT-XL under the same setting. We remain behind PixelGen-XL, which reaches 1.83 at 160 epochs in a similar parameter range; we attribute this to its use of frozen VGG (Simonyan & Zisserman, 2014) supervision alongside DINO, and leave the combination of the two signals to future work.

5.3 DINO inversion

Our method assumes the branch produces something semantically meaningful, not only an arbitrary code that happens to help the velocity prediction. We visualise the PCA of the predicted features across the trajectory in Fig. 5 and Fig. 14): their structures are spatially coherent and separates object from background before the image itself is resolved, and it sharpens as denoising progresses.

Table 6: Unconditional generation on ImageNet 256^2. (no CFG, 80 epochs).

Model	#Params	Steps	FID↓
SiT-L/2	458M	400K	49.51
→ **DF**	466M	400K	**9.14**
JiT-L/16	459M	100K	53.97
→ LatentForcing	465M	100K	20.44
→ **DF**	468M	100K	**15.19**

Table 7: Effect of the DF supervision target on SiT-L/2 at 80 epochs.

Model	#Params	FID↓
Latent space		
SiT-L/2	458M	18.80
→ DF, no supervision	466M	12.96
→ DF, image supervision	466M	11.93
→ **DF, DINO supervision**	466M	**3.04**

Furthermore, as the representation-conditioning branch processes representation features, we can bypass the prediction at inference and feed real DINOv2 features instead. This turns the model into a DINOv2 inverter: given the features of an image and a null class, it reconstructs the image itself. Figure 6 shows reconstructions from inverted DINOv2. Table 5 reports SSIM and PSNR for SiT (16.86 dB / 0.48 SSIM) and JiT (14.33 dB / 0.44 SSIM), showing that the interaction of the DINOv2 condition provides very good local information for reconstruction.

5.4 DINO ONLY, NO CLASS

The predicted DINOv2 from the representation-conditioning branch can be seen as a standalone condition. To test this, we train with the class token set to null for the entire run, so the model has no conditioning signal other than the one it predicts for itself. We compare this setup with unconditional SiT and JiT and with Latent Forcing (Baade et al., 2026) in Table 6. In latent space, **DF** reaches an FID of 9.14 against 49.51 for SiT-L/2. In pixel space it reaches 15.19, against 53.97 for JiT-L/16 and 20.44 for Latent Forcing. This setting isolates the main difference with Latent Forcing: when all of the conditioning comes from the pretrained representation branch, predicting it outperforms denoising it on a fixed schedule.

5.5 ABLATION

We ablate design choices of DF; more ablations in Appendix C. Unless stated otherwise, every ablation is run on SiT-L and reported in the unguided setting, so that all numbers are directly comparable.

Does the pretrained representation target matter? The improvement from **DF** could come from the conditioning mechanism, or from the pretrained representation content of the target. To separate the two, we train three variants with identical architectures that differ only in what the representation branch is asked to predict (Table 7). DF, no supervision leaves the predicted features unsupervised. DF, image supervision supervises the branch to predict the image itself, a well-defined but low-level target. DF, DINO supervision uses DINOv2 features, as in our full method. The conditioning mechanism helps on its own: without any supervision, FID drops from 18.80 to 12.96. Supervising with the raw image adds little (11.93). So a reconstruction target is not what drives the gain. With DINOv2 features, FID reaches 3.04, 6× below the baseline. The conditioning mechanism is necessary but not sufficient; the pretrained representation content of the target is what produces most of the improvement.

6 CONCLUSIONS

We introduced DINO-Forcing, a simple alternative to co-denoising pretrained representations in flow-matching models. Instead of assigning the representation its own noisy trajectory, we predict it from the current image state and immediately reuse it as conditioning. This removes the second ODE and representation-specific denoising schedule, while preserving the benefits of representation guidance. Despite its simplicity, this change substantially improves both convergence and generation quality, surpassing methods trained for substantially longer both in latent and pixel space. More broadly, our results suggest that auxiliary representations need not always be generated alongside the primary modality: when they can be inferred from the evolving sample, prediction may be enough.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, and the rest of the required disclosure tasks (generate synthetic data sets, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results) are not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures or images, creation of artifacts, draft parts of a research paper, brainstorming, sourcing/searching for information, edit a research paper to improve readability, identify relevant literature, and format references.

We have reviewed all AI-assisted work. We checked LLM-generated code, LLM-generated suggestions have been tested and backed-up by experimental results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Li Fei-Fei. Latent forcing: Reordering the diffusion trajectory for pixel-space image generation. *arXiv arXiv:2602.11401*, 2026.

Fan Bao, Shen Nie, Kaiwen Xue, Chongxuan Li, Shi Pu, Yaole Wang, Gang Yue, Yue Cao, Hang Su, and Jun Zhu. One transformer fits all distributions in multi-modal diffusion at scale. In *ICML*, pp. 1692–1717. PMLR, 2023.

Hun Chang, Byunghee Cha, and Jong Chul Ye. Dino-sae: Dino spherical autoencoder for high-fidelity image reconstruction and generation. *arXiv arXiv:2601.22904*, 2026.

Ting Chen, Ruixiang ZHANG, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *ICLR*, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Lucas Degeorge, Paul Couairon, Arijit Ghosh, Alexei A Efros, David Picard, and Vicky Kalogeiton. Balancing frequencies and pixels in flow matching. *arXiv preprint arXiv:2609.02748*, 2026.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Nicolas Dufour, Victor Besnier, Vicky Kalogeiton, and David Picard. Don't drop your samples! coherence-aware training benefits conditional diffusion. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6264–6273. IEEE, 2024.

Karl Heun et al. Neue methoden zur approximativen integration der differentialgleichungen einer unabhängigen veränderlichen. *Z. Math. Phys*, 45(23-38):7, 1900.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

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

Allan Jabri, David J Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In *ICML*, pp. 14569–14589. PMLR, 2023.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. *Advances in Neural Information Processing Systems*, 37:52996–53021, 2024.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Theodoros Kouzelis, Spyros Gidaris, and Nikos Komodakis. Coevolving representations in joint image-feature diffusion. *arXiv arXiv:2604.17492*, 2026a.

Theodoros Kouzelis, Efstathios Karypidis, Ioannis Kakogeorgiou, Spyridon Gidaris, and Nikos Komodakis. Boosting generative image modeling via joint image-feature synthesis. *NeurIPS*, 38: 16685–16714, 2026b.

Felix Krause, Timy Phan, Ming Gui, Stefan Andreas Baumann, Vincent Tao Hu, and Björn Ommer. Tread: Token routing for efficient architecture-agnostic diffusion training. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15703–15713. IEEE, 2025.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. *Advances in Neural Information Processing Systems*, 37:122458–122483, 2024.

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

Tianhong Li, Dina Katabi, and Kaiming He. Return of unconditional generation: A self-supervised representation generation method. *Advances in Neural Information Processing Systems*, 37: 125441–125468, 2024.

Han Lin, Xichen Pan, Zun Wang, Yue Zhang, Chu Wang, Jaemin Cho, and Mohit Bansal. V-co: A closer look at visual representation alignment via co-denoising. *arXiv arXiv:2603.16792*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, pp. 23–40. Springer, 2024.

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

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

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

Yueming Pan, Ruoyu Feng, Qi Dai, Yuqi Wang, Wenfeng Lin, Mingyu Guo, Chong Luo, and Nanning Zheng. Semantics lead the way: Harmonizing semantic and texture modeling with asynchronous latent diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 43664–43674, 2026.

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

Giorgos Petsangourakis, Christos Sgouropoulos, Bill Psomas, Theodoros Giannakopoulos, Giorgos Sfikas, and Ioannis Kakogeorgiou. Reglue your latents with global and local semantics for entangled diffusion. *arXiv arXiv:2512.16636*, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

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

Jaeyo Shin, Jiwook Kim, and Hyunjung Shim. Representation alignment for just image transformers is not easier than you think. *arXiv preprint arXiv:2603.14366*, 2026.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Jaskirat Singh, Xingjian Leng, Zongze Wu, Liang Zheng, Richard Zhang, Eli Shechtman, and Saining Xie. What matters for representation alignment: Global information or spatial structure? In *ICLR*, 2026a. URL https://openreview.net/forum?id=y0UxFtXqXf.

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

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

Chenyu Wang, Cai Zhou, Sharut Gupta, Zongyu Lin, Stefanie Jegelka, Stephen Bates, and Tommi Jaakkola. Learning diffusion models with flexible representation guidance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=cIGfKdfy3N.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 40633–40642, 2026.

Ziqiao Wang, Wangbo Zhao, Yuhao Zhou, Zekai Li, Zhiyuan Liang, Mingjia Shi, Xuanlei Zhao, Pengfei Zhou, Kaipeng Zhang, Zhangyang Wang, Kai Wang, and Yang You. REPA works until it doesn't: Early-stopped, holistic alignment supercharges diffusion training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=HK96GI5s7G.

Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Ming-Ming Cheng, et al. Representation entanglement for generation: Training diffusion transformers is much easier than you think. *NeurIPS*, 38:7714–7743, 2026.

Xingqian Xu, Zhangyang Wang, Gong Zhang, Kai Wang, and Humphrey Shi. Versatile diffusion: Text, images and variations all in one diffusion model. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7754–7765, 2023.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *ICLR*, 2025. URL https://openreview.net/forum?id=DJSZGGZYVi.

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

Yongsheng Yu, Wei Xiong, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit2: Representation-grounded pixel diffusion transformers, 2026b. URL https://arxiv.org/abs/2609.24919.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *ICLR*, 2026. URL https://openreview.net/forum?id=0u1LigJaab.

Xingyu Zhou, Qifan Li, Xiaobin Hu, Hai Chen, and Shuhang Gu. Guiding a diffusion transformer with the internal dynamics of itself. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11536–11545, 2026.

APPENDIX

A IMPLEMENTATION DETAILS

Table 8: Training and sampling configuration of **DF** in latent and pixel space on ImageNet.

	Latent space	**Pixel space**
Backbone		
Base implementation	`SiT` (Ma et al., 2024)	`JiT` (Li & He, 2026)
Input space	`sd-vae-ft-ema` latents	RGB pixels
Resolution	256×256, 512×512	256×256
Patch size	2	16
Dropout (XL only)	0.1, middle third of blocks	0.1, middle third of blocks
Representation Branch		
Target encoder ϕ	`DINOv2-B`	`DINOv2-B`
Representation dim. D'	768	768
Fusion	channel concat. + linear	channel concat. + linear
Prediction block index	last	last
Prediction block type	`REPA MLP`	`REPA MLP`
$\mathcal{L}_{\text{FR}}$	cosine similarity	cosine similarity
Loss weight λ	0.5	0.03
Optimization		
Optimizer	Adam	Adam
Learning rate	1×10^{-4} (constant)	2×10^{-4} (constant)
Warmup	–	5 epochs
Weight decay	0	0
Batch size	256	1024
EMA decay	0.9999	0.9999
Training epochs	80 (`B`, `L`) / 100 (`XL`)	80 (`B`) / 200 (`L`, `XL`)
Conditioning Dropout		
Null class (CFG)	10%	10%
Pass 1 only (p)	10%	10%
Ground-truth $\phi(x)$ as condition	5%	5%
Sampling		
Solver	Euler (ODE)	Heun (ODE)
Steps	50	50
Guidance	AutoGuidance (`L` @ 150K)	CFG + interval (0.1,1)
CFG sweep range	(1.0, 4.0)	(1.0, 4.0)
CFG sweep range interval	0.1	0.1

B EXPLANATION BEHIND OUR FUSING IMAGE AND FROZEN REPRESENTATIONS MECHANISM

One particularly interesting question that arises from Figure 3 is how we *merge* the image and the frozen representations. Given we use `DiT` based transformer architecture, we use same number of tokens as in the image. As such, any token-wise merge operation suffices. However, for our experiments, we fuse our image and representation tokens by concatenating them via the channel dimension. Given an image (or its feature), x, we extract patch tokens of shape `(B,T,D)` via standard patchification. We set our starting representation token dimension to `(B,T,D')` matching the dimension of frozen pretrained encoder (e.g., `DINOv2` (Oquab et al., 2024)). The two token sequences are thereafter spatially aligned and concatenated along the channel dimension, yielding a fused representation of shape `(B,T,D+D')`, which is then projected to the transformer's hidden dimension via a lightweight linear layer. This keeps the token count identical to a standard image-only model and requires no spatial resampling, making it directly compatible with existing flow matching architectures. Our fusion strategy is similar in spirit to concurrent work (Kouzelis et al., 2026b), but simpler by virtue of requiring no upsampling. There are many other ways we can fuse the tokens,

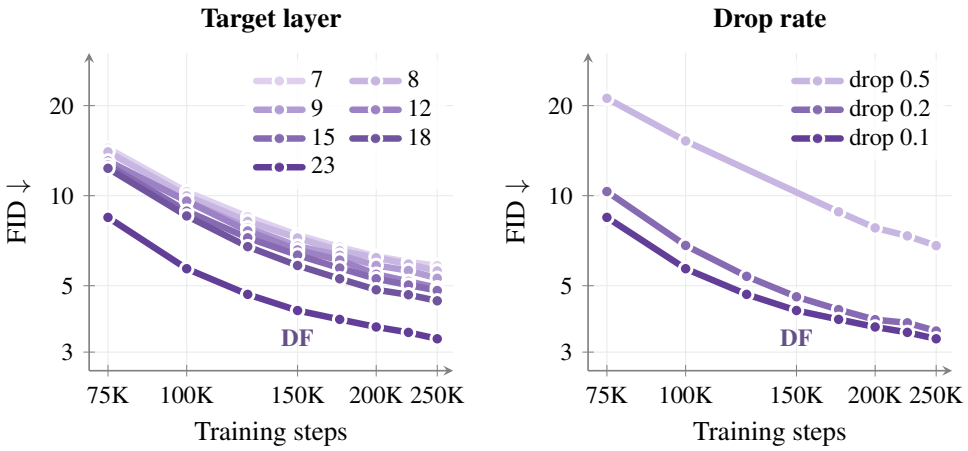

Figure 7: Layer index and dropping probability ablation on SiT-L model.

and we urge the readers to follow the excellent `V-Co` (Lin et al., 2026) paper, which does a deep dive into different fusion techniques. Here, we stick to the channel concatenation strategy just for proper comparison and leave the exploration of other types of fusion strategies for future work.

C Ablation

We now ablate the main design choices of our method. Unless stated otherwise, every ablation is run on `SiT-L` and reported in the unguided setting, so that all numbers are directly comparable.

Where should the pretrained representation live? We sweep the block at which the pretrained representation prediction is attached and plot the resulting FID trajectories in Figure 7 (left). Deeper is consistently better throughout training: the final block (`Layer 23`, zero-indexed) reaches 3.32 against roughly 5.84 for `Layer 7` at 250K steps. This is consistent with the branch requiring the abstract, high-level features that emerge only deep in the network, which are also the features that align with the `DINOv2` target. We attach the branch at the deepest block in all main experiments.

How often should we train on Pass 1 alone? Pass 1 carries a specific burden: at sampling time it is the first step the model takes and its look-ahead must orient the trajectory before any real conditioning exists. Pass 1 is optimized alone with probability p, We sweep it in Figure 7 (right) this probability. $p = 0.1$ works best (3.32) compared to $p = 0.2$ (3.53) and $p = 0.5$ (6.81) at 250K steps. We use $p = 0.1$ throughout.

D Extended related work

Representation alignment in image generation. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020) and flow matching models (Lipman et al., 2023) are trained with denoising objectives that predominantly encourage fine-grained, reconstruction-oriented internal representations rather than high-level semantic ones (Yu et al., 2025). This lack of semantic structure slows training convergence with state-of-the-art image generation models typically requiring over 500 training epochs to reach strong performance (`DiT`: Peebles & Xie, 2023; `SiT`: Ma et al., 2024; `JiT`: Li & He, 2026; fv`-loss`: Degeorge et al., 2026; `TREAD`: Krause et al., 2025). Recent works (`REPA`: Yu et al., 2025; `iREPA`: Singh et al., 2026a; `HASTE`: Wang et al., 2025b; `REED`: Wang et al., 2025a; `DDT`: Wang et al., 2026; `PixelDiT`: Yu et al., 2026a; `PixelREPA`: Shin et al., 2026; `DeCo`: Ma et al., 2026a; `PixelGen`: Ma et al., 2026b) have shown that explicitly aligning a model's internal features with external pretrained representations substantially accelerates convergence. A complementary line of work takes this further by performing flow matching directly in a pre-aligned representation latent space (`RAE`: Zheng et al., 2026; `RAEv2`: Singh et al., 2026b; `DINO-SAE`: Chang et al., 2026), yielding additional convergence speedups. Despite their promise, both approaches have their own limitations

when applied to image generation. Feature alignment methods inject pretrained information only during training; at sampling time, the model operates without any explicit guidance, effectively discarding the alignment signal precisely when it is needed. Pre-aligned latent approaches sidestep this by embedding pretrained information into the generation space itself, but at the cost of training a new decoder and introducing non-trivial architectural modifications. A natural middle ground is to jointly denoise both pretrained representation and image, which we discuss next.

Co-denoising for image generation. Jointly denoising multiple modalities (Bao et al., 2023; Xu et al., 2023) offers an appealing alternative: it implicitly aligns pretrained representations while preserving them throughout the entire generation process, including at inference. Applied to image generation, this strategy accelerates training convergence, training both in a latent image feature space (`ReDi`: Kouzelis et al., 2026b; `CoReDi`: Kouzelis et al., 2026a; `REG`: Wu et al., 2026; `SFD`: Pan et al., 2026 `REGLUE`: Petsangourakis et al., 2025) or directly in pixel space (`LatentForcing`: Baade et al., 2026; `V-Co`: Lin et al., 2026; `CoReDi`: Kouzelis et al., 2026a). However, co-denoising introduces its own complexity: the optimal noise schedule for the pretrained representation is non-trivial to determine and must be carefully hand-crafted. For instance, `SFD` (Pan et al., 2026) and `LatentForcing` (Baade et al., 2026) found that denoising the pretrained representation branch (e.g., `DINO` features) to a low noise level *before* beginning image denoising yields the best convergence speed, a schedule that likely needs to be re-tuned for different modalities, making the approach brittle to apply in general.

The limitations above share a common structure: they all stem from either the absence of pretrained information during image generation, or the need for additional models and carefully designed schedules to introduce it. One principled way to address this is to condition the image generation model directly on pretrained representations, bypassing the need to denoise them from scratch (`RCG`: Li et al., 2024; `PixelDiT2`: Yu et al., 2026b). Inspired by self-conditioning techniques (Jabri et al., 2023; Chen et al., 2023), we propose to condition the model on its *own predicted representation* at each denoising step. Crucially, unlike prior self-conditioning works, we explicitly make these predicted self-conditions to align with an external pretrained feature space, encouraging the model to *predict its own representation* in a meaningful, structured sense. This design simultaneously resolves the key failure modes of existing approaches: pretrained information is available throughout image generation without requiring a separate decoder, no architectural modifications, or handcrafted denoising schedules, while still achieving faster training convergence.

E ADDITIONAL QUALITATIVE RESULTS

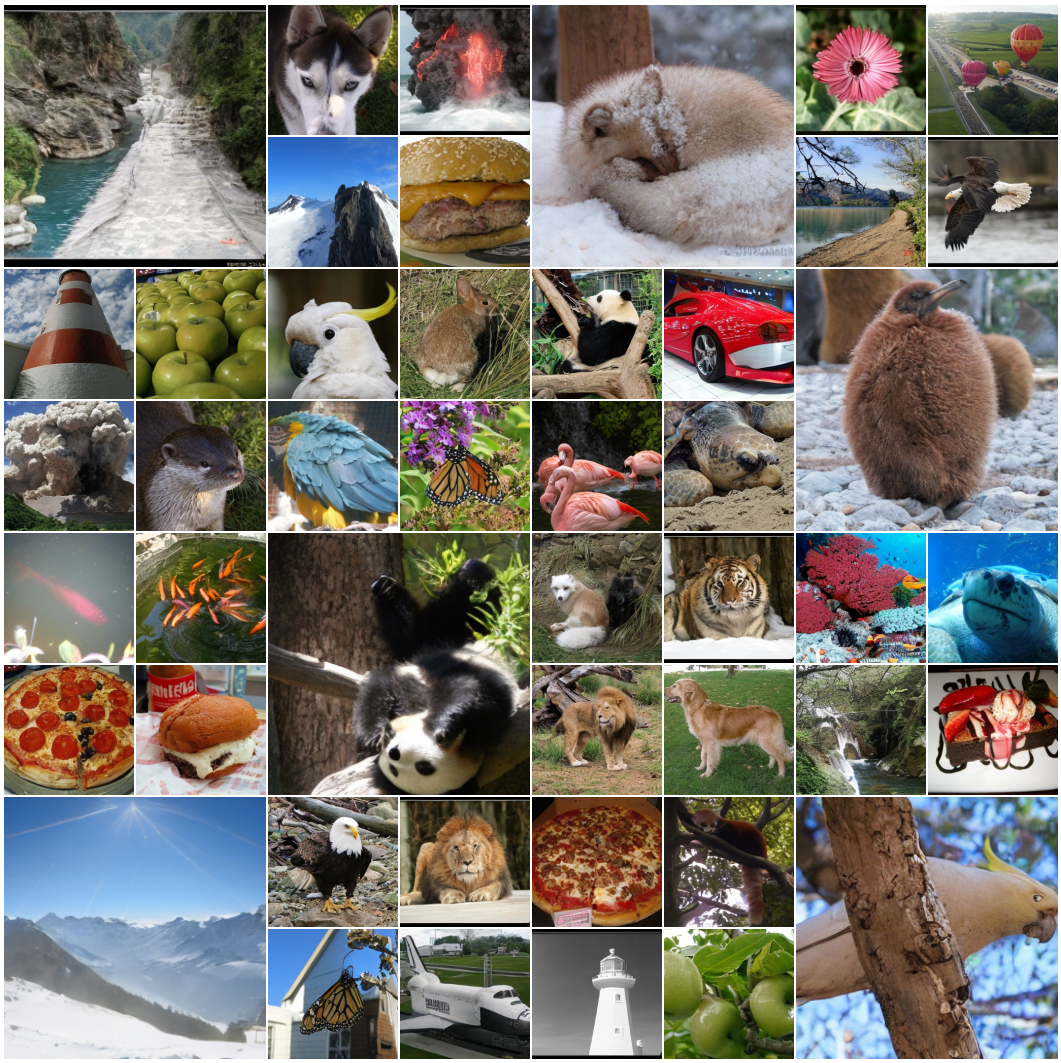

Figure 8: Uncurated samples from DF-SiT-XL/2 on ImageNet 256×256, without guidance.



Figure 9: Uncurated samples from DF-SiT-XL/2 on ImageNet 256×256, with cfg=4.0.

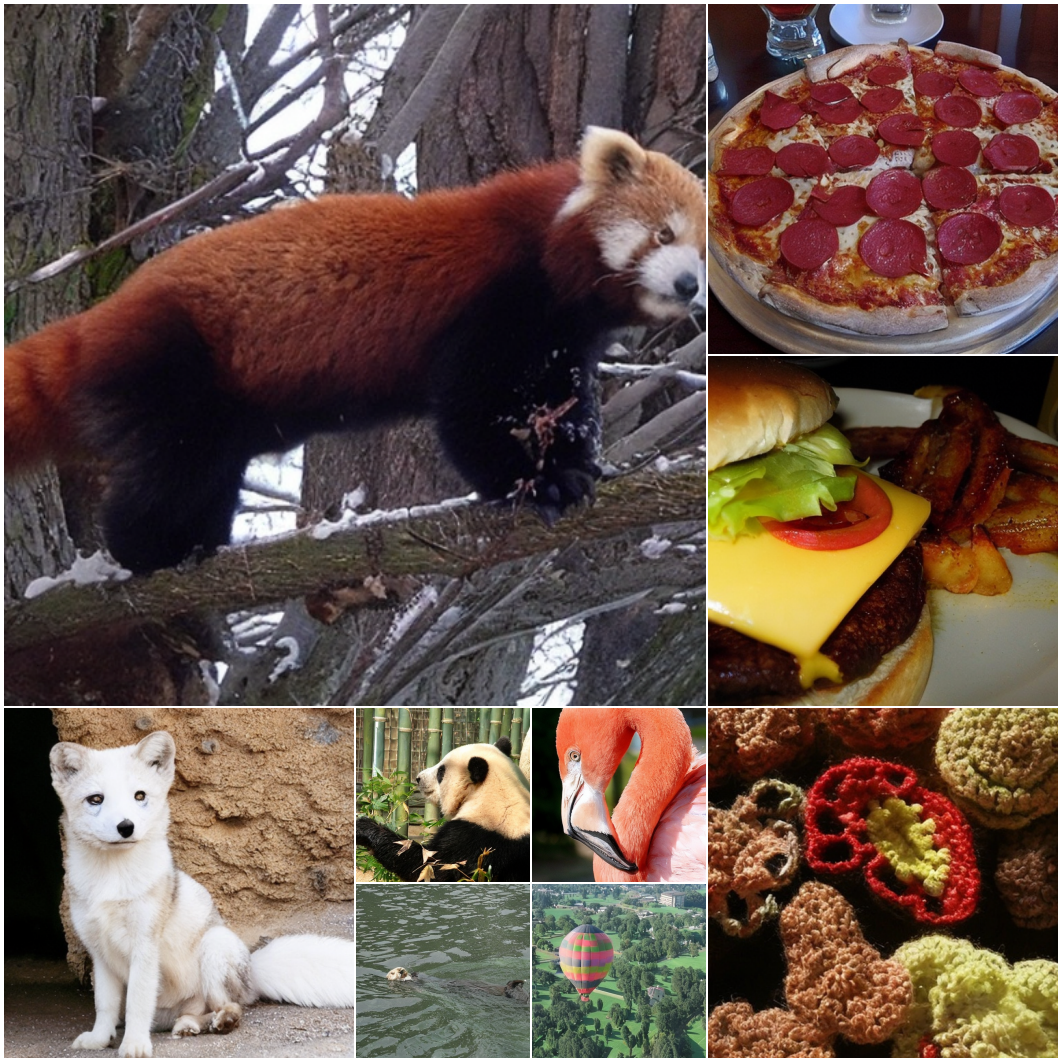

Figure 10: Uncurated samples from DF-SiT-XL/2 on ImageNet 512×512 without guidance.



Figure 11: Uncurated samples from DF-SiT-XL/2 on ImageNet 512×512, with cfg=4.0.

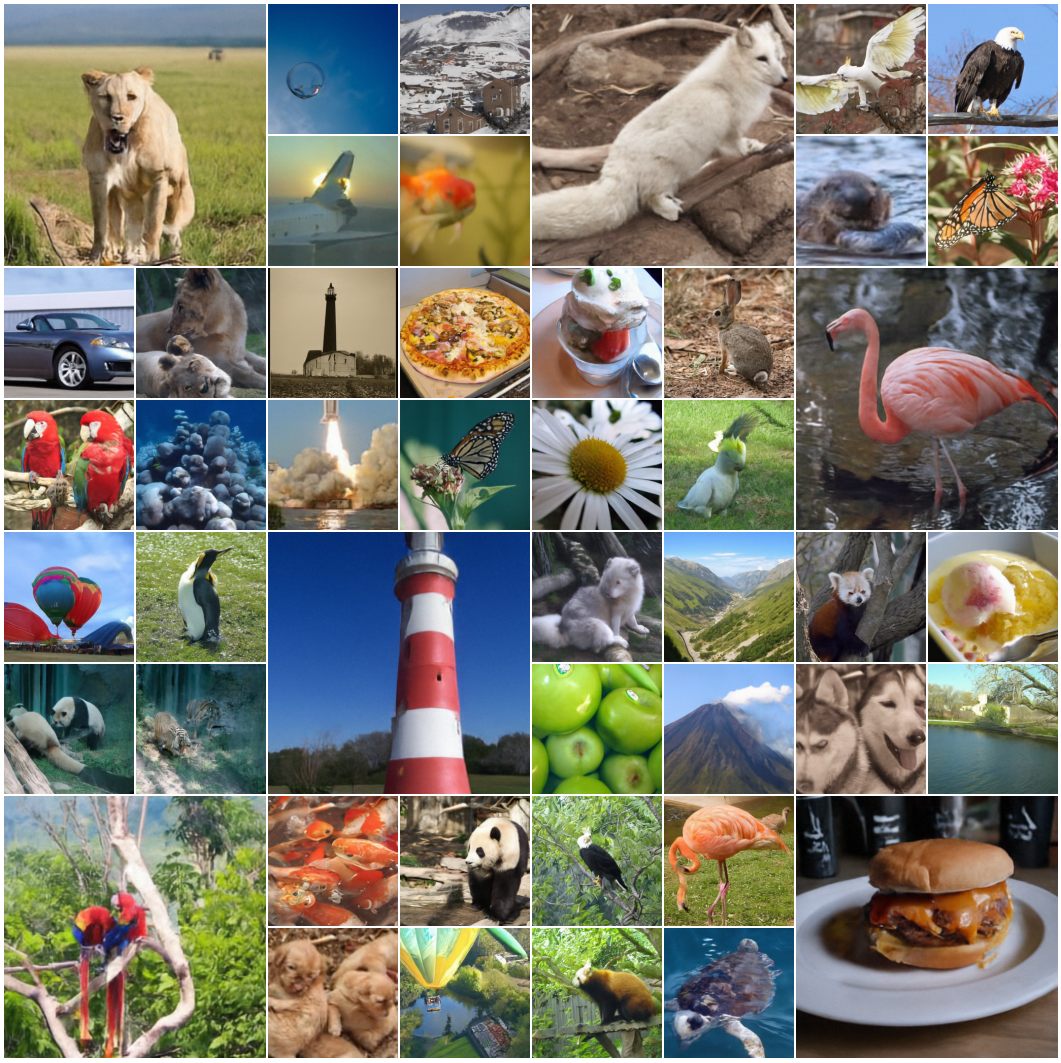

Figure 12: Uncurated samples from DF-JiT-XL/16 on ImageNet 256×256 without guidance.



Figure 13: Uncurated samples from DF-JiT-XL/16 on ImageNet 256×256 with cfg=4.0.

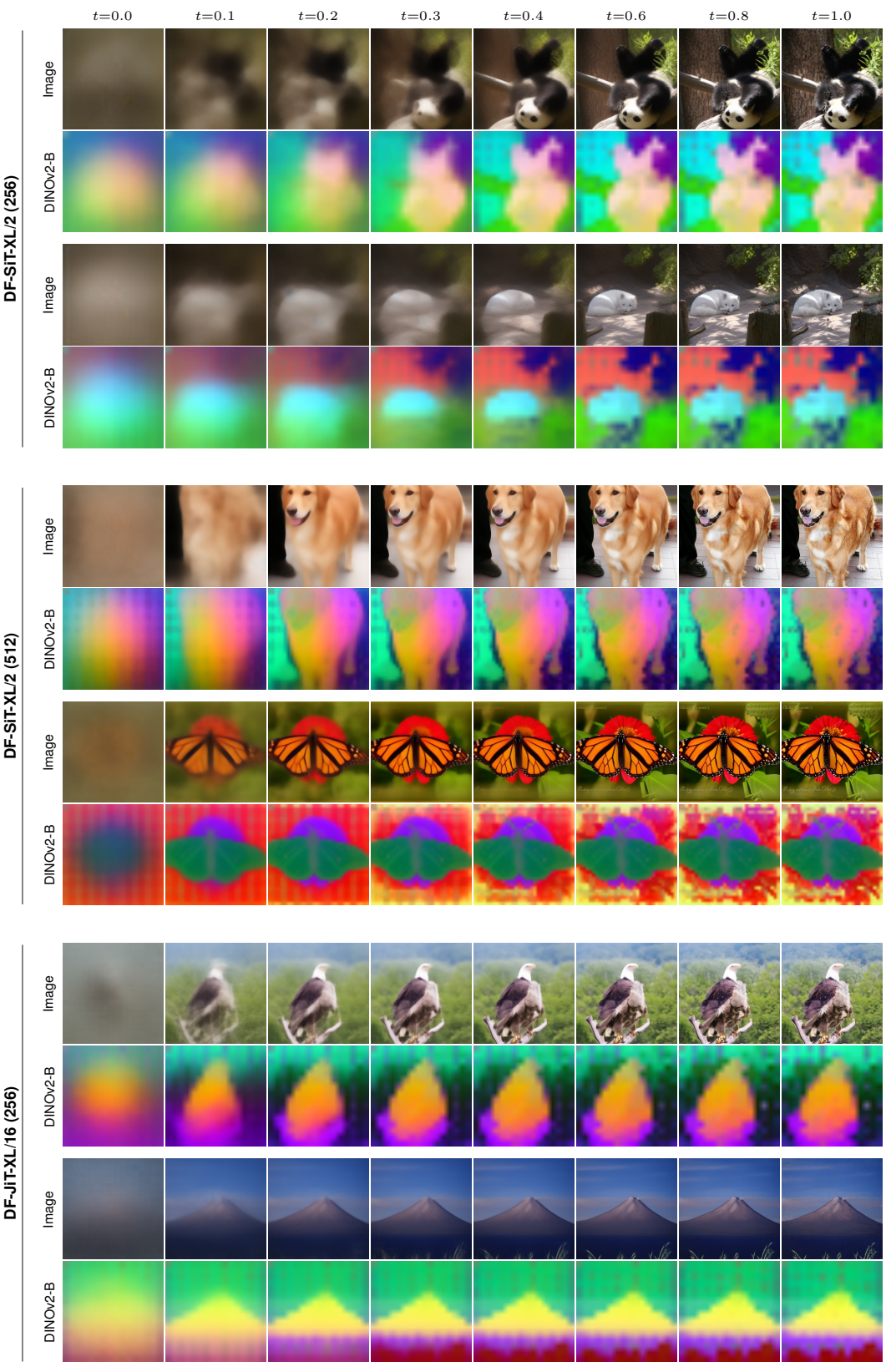

Figure 14: Evolution of the image and the PCA of the predicted DINO features throughout sampling (t=0: noise, t=1: data), for DF-SiT-XL/2 at 256×256 and 512×512 and DF-JiT at 256×256. The PCA projection is fit once per sample on the final features and reused for all timesteps.