Trajectory Forcing: Structure-First Generation with Controllable Semantic Trajectories

Merve Kocabas[1,4], Gege Gao[1,3]†,
Bernhard Schölkopf[3,4,5,6]‡, and Andreas Geiger[1,2,5,6]‡

[1] University of Tübingen, Tübingen, Germany
[2] Tübingen AI Center, Tübingen, Germany
merve.kocabas@student.uni-tuebingen.de
{gege.gao,a.geiger}@uni-tuebingen.de
[3] ETH Zürich, Zürich, Switzerland
[4] Max Planck Institute for Intelligent Systems, Tübingen, Germany
bs@tuebingen.mpg.de
[5] ELLIS Institute, Tübingen, Germany
[6] KE:SAI, Tübingen, Germany

† Project lead. ‡ Shared last authorship.

https://mervekocabas.github.io/TrajectoryForcing/

Abstract. Diffusion and flow-based generative models produce strong images, yet their controllability remains largely endpoint-centric: users specify conditions and receive final outputs, while the intermediate generative dynamics remain hidden. Recent methods have begun to exploit generation order and process decomposition to improve sample quality, but still treat intermediate states as internal computation rather than objects for interaction. We propose **Trajectory Forcing (TF)**, a trajectory-centric framework that makes the generation path explicit, semantic, and editable. TF organizes synthesis as a sequence of semantically structured stages, progressing from global layout to object-, part-, and detail-level representations. Each stage produces a decodable latent state that can be inspected, evaluated, and locally edited before the next stage begins. To instantiate this path, we derive coarse-to-fine teacher hierarchies by clustering pretrained visual representations such as DINOv2, and train a hierarchy-conditioned one-step flow-matching model at each level. We further introduce trajectory-aware metrics that measure structural consistency and local controllability beyond endpoint quality metrics such as FID. Experiments show that TF achieves competitive sample quality while exposing coherent intermediate states and supporting localized edits across semantic levels. By shifting the focus from final images to the generative path itself, TF opens a route toward controllable, trajectory-aware image synthesis.

Keywords: Trajectory-Centric Generation · Hierarchical Intermediate States · Structured Generative Control

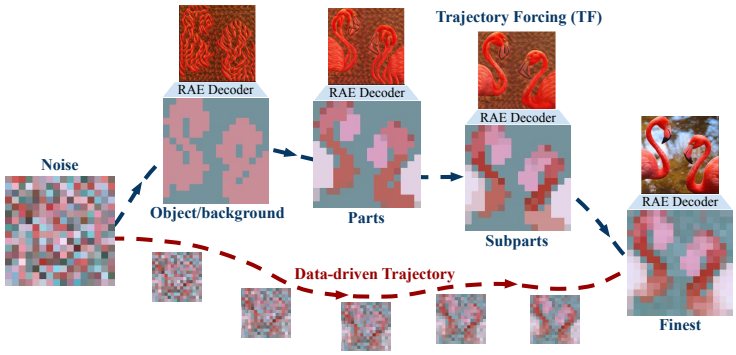


Fig. 1: Trajectory Forcing (TF) replaces the opaque, data-driven denoising trajectory (bottom, red) with a deliberately structured coarse-to-fine progression through semantic levels (top, blue). Every intermediate state is decodable via a shared decoder into a visually interpretable image, enabling inspection and editing at each level.

1 Introduction

To control generation, we must expose not only its destination, but also its path.

Modern image generators typically cast synthesis as a direct mapping from noise to a finished image. Although denoising traverses intermediate states, these states are not deliberately structured for human understanding or intervention; they are computational by-products discarded after the final output is produced. This contrasts with human visual creation, which is inherently coarse-to-fine: global composition is established before local details are refined, and each intermediate stage can be inspected and revised. We take this contrast as our starting point and ask whether generative trajectories themselves can be made interpretable, semantically structured, and editable.

Recent works have begun to exploit trajectory structure in generation [2,4,28, 41], showing that generation order and process decomposition can substantially affect output quality. However, these trajectories remain optimized for the final image: intermediate states are used as computational scratchpads, frequency components, or token-completion steps, rather than as semantic objects a user can inspect, compare, and redirect.

We therefore propose **Trajectory Forcing (TF)**, a framework that organizes image synthesis as a sequence of semantically structured stages. Each stage produces an interpretable latent state that can be decoded, evaluated, and edited before the next stage begins. Rather than collapsing generation into a single opaque mapping from noise to image, TF exposes a coarse-to-fine hierarchy from global layout, through object-level structure, to fine-grained detail, making every intermediate level visually meaningful and amenable to controlled intervention.

Making this hierarchy operational requires a representation space whose geometry organizes semantic structure. Raw pixels entangle appearance, layout, and identity, whereas pretrained visual representations provide a more suitable substrate. Following the *representation alignment hypothesis* [45], we operate in DINOv2 feature space [30], where hierarchical clustering over latent tokens em-

pirically recovers object-part-subpart decompositions (Sec. 4.1) that supervise our trajectory design.

A natural concern with multi-stage generation is inference cost: progressive formulations typically require network function evaluations (NFE) that scale as $L \times T$, where L is the number of structural stages and T the number of denoising steps per stage. TF addresses this by integrating one-step flow matching [13,14, 26] at each stage. Instead of iterative denoising at every level, each stage uses a single function evaluation to map noise to the current-level target, conditioned on the previous stage. This reduces inference to L-NFE, making TF comparable to modern few-step generators while preserving trajectory-level controllability.

Our contributions are as follows:

- We propose **Trajectory Forcing (TF)**, a generation framework that models image synthesis as a structured coarse-to-fine trajectory of interpretable semantic stages, enabling inspection and editing at every level.
- We introduce a **hierarchy-conditioned one-step flow** formulation that extends one-step generation to multi-level trajectories via level-wise conditioning and structural consistency supervision.
- We design **trajectory-aware metrics** that go beyond FID to measure structural consistency and local controllability across generation levels.
- We validate TF on class-conditional ImageNet generation, achieving competitive image quality and fast FID convergence while enabling interpretable multi-level trajectories and interactive editing.

2 Related Work

2.1 Diffusion and Flow-Based Generation

Latent-space models. Latent Diffusion Models [33] established the paradigm of performing generative dynamics in a compressed latent space, decoupling representation learning from the denoising process. Subsequent work scales the backbone with Transformers [27, 31] or revisits the representation interface through alternative autoencoding schemes [47].

Pixel-space models. A parallel line of work revisits pixel-space generation, removing reliance on pretrained tokenizers. Recent approaches include flow-based pixel modeling [5], neural-field diffusion [40], and end-to-end training with self-supervised pretraining [20]. JiT [21] shows that direct clean-image prediction enables diffusion in high-dimensional pixel spaces, establishing a strong pixel-space baseline. While these works show that compression-based latents are not strictly necessary, our use of a representation space is motivated by semantic structure rather than compression: pretrained features support both hierarchy construction and visual decodability of intermediate states.

One and few-step models. Reducing inference cost is a major research direction [17, 42, 46]. Distillation-based methods compress multi-step models [29, 35]. Consistency models learn to map arbitrary intermediate states to the ODE endpoint [25, 36, 37], with trajectory-aware extensions [19]. More recently, shortcut parameterizations [9, 48] and reparameterized matching objectives, including

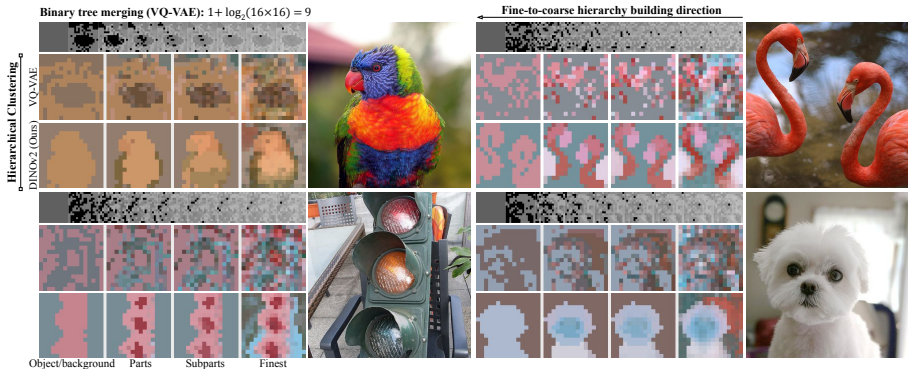


Fig. 2: Hierarchy comparison. NVG's binary merging in VQ-VAE latent space produces a resolution-tied hierarchy ($1+\log_2(16\times16) = 9$ levels) whose intermediate stages lack clear semantic groupings. In contrast, clustering DINOv2 features recovers a compact object-part-subpart hierarchy with semantically interpretable levels.

Mean Flows [13, 14, 26] and drifting-based formulations [7], enable one-step or few-step generation. Our work inherits the mean-flow framework and extends it with hierarchical conditioning and structural supervision.

2.2 Generation Ordering and Trajectory Design

The ordering of generation has recently emerged as an important design axis for controlling *how information is revealed* during synthesis.

Per-token noise scheduling. Diffusion Forcing [4] assigns independent noise levels to individual tokens, enabling flexible generation orderings in sequential data. This idea is theoretically grounded: different conditioning orders can lead to exponential differences in the learnability of data distributions [15].

Frequency-domain reordering. DeCo [28] factorizes pixel diffusion along frequency components, using a lightweight decoder for high-frequency details while the main DiT specializes in low-frequency semantics. This decoupling improves both training efficiency and generation quality for pixel-space models.

Latent-pixel reordering. Latent Forcing [2] jointly diffuses self-supervised latent features and pixels with separate time schedules. By revealing latent structure before pixel detail, it achieves the convergence benefits of latent diffusion while remaining end-to-end in pixel space. The generated latent features serve as a computational scratchpad and are discarded after generation.

All of these methods use trajectory structure to improve *generation quality*, while intermediate states remain either implicit or disposable. In contrast, our work treats the structured trajectory as a user-facing interface: intermediate stages are designed to be semantically interpretable, decodable, and editable, enabling trajectory-level control that goes beyond endpoint optimization.

2.3 Progressive and Hierarchical Generation

Autoregressive and masked models. Token-based approaches generate images by predicting discrete tokens [8] or masked subsets [22], sometimes combined with diffusion-based training [23, 43]. These naturally expose partial intermediate

samples, but progression is defined over token completion rather than continuous semantic refinement.

Scale-level progression. Visual Autoregressive Modeling (VAR) [38] and FlowAR [32] organize generation along spatial resolution, predicting coarse structures first and refining at higher resolutions. Our hierarchy is orthogonal: levels correspond to semantic granularity (object, part, subpart) at a fixed spatial resolution, rather than resolution upscaling.

Granularity-level progression. Explicit semantic or object-level structure has been used to organize visual scenes in related settings [1,10–12,39]. Most closely related to our hierarchical design, Next Visual Granularity (NVG) [41] decomposes images into granularity levels by bottom-up clustering in VQ-VAE latent space, and autoregressively predicts structure maps and tokens at each level. While NVG demonstrates the value of explicit structure control, its hierarchy is constrained by greedy binary L2 merging ($k = 2$) in VQ-VAE space, whose limited semantic geometry makes meaningful groupings, such as object-part-subpart, difficult. Consequently, its stages are tied to latent resolution (e.g., $\log_2(16\times16) = 8$), its intermediates lack clear semantic abstractions (Fig. 2), and its discrete codebook limits intermediate-state flexibility and continuity.

Our approach shares NVG's goal of making structure explicit in generation, but differs in substrate, dynamics, and interface: we build semantically grounded hierarchies by clustering geometrically regular pretrained representations (e.g., DINOv2 [30]), use one-step flow matching per level for L-NFE inference instead of $L\times T$, and keep all intermediate states continuously decodable and editable for interactive trajectory-level control.

2.4 Representation-Space Generation

Recent work has shown that pretrained visual representations can improve generative modeling. REPA [45] aligns intermediate diffusion features with DINOv2 [30] latents via an auxiliary loss, accelerating convergence. Representation Autoencoders (RAE) [47] further train decoders to reconstruct images from DINOv2 features, enabling generation in a semantically structured and visually decodable representation space. This provides the two ingredients our framework relies on: a geometry suitable for coarse-to-fine hierarchies, and a decoder that makes each intermediate level both semantic and visually interpretable.

3 Background

We build on two technical ingredients: (i) a flow-matching formulation that enables one-step generation, and (ii) a pretrained representation space whose geometric structure supports both meaningful hierarchical decomposition and visual decoding of intermediate states.

3.1 Flow Matching and Mean Flows

We briefly review the progression from flow matching to one-step mean-flow generators, culminating in the backbone we build upon.

Flow Matching. Flow Matching [24] learns a velocity field that transports between a prior distribution and the data distribution. Given data $\mathbf{x} \sim p_{\text{data}}$

and noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, a flow path is defined by the linear interpolation

$$\mathbf{z}_t = (1-t)\,\mathbf{x} + t\,\boldsymbol{\epsilon}, \tag{1}$$

with $t \in [0, 1]$. The conditional velocity is $\mathbf{v} = \boldsymbol{\epsilon} - \mathbf{x}$, and a network $\mathbf{v}_\theta$ is trained by minimizing $\mathcal{L}_{\text{FM}} = \mathbb{E}_{t,\mathbf{x},\boldsymbol{\epsilon}} \|\mathbf{v}_\theta(\mathbf{z}_t, t) - \mathbf{v}\|^2$. Samples are generated by solving the ODE $\frac{d}{dt}\mathbf{z}_t = \mathbf{v}_\theta(\mathbf{z}_t, t)$ from t=1 (noise) to t=0 (data), typically requiring many network evaluations.

Mean Flows. MeanFlow (MF) [13] enables one-step generation by modeling an *average velocity* field. Viewing Flow Matching's $\mathbf{v}(\mathbf{z}_t, t)$ as the *instantaneous* velocity, MF defines the average velocity over an interval $[r, t]$ as:

$$\mathbf{u}(\mathbf{z}_t, r, t) \triangleq \frac{1}{t-r} \int_r^t \mathbf{v}(\mathbf{z}_\tau, \tau)\, d\tau. \tag{2}$$

Since directly evaluating this integral during training is intractable, MF differentiates both sides with respect to t to derive the *MeanFlow Identity*:

$$\mathbf{u}(\mathbf{z}_t, r, t) = \mathbf{v}(\mathbf{z}_t, t) - (t-r)\,\frac{d}{dt}\mathbf{u}(\mathbf{z}_t, r, t), \tag{3}$$

where the total derivative $\frac{d}{dt}\mathbf{u}$ is computed via a Jacobian-vector product (JVP). A network $\mathbf{u}_\theta(\mathbf{z}_t, r, t)$ is trained to satisfy this identity using the conditional velocity $\mathbf{v}(\mathbf{z}_t, t)$ as ground-truth signal and a stop-gradient on the JVP term, denoted JVP_{sg} [13]. Once trained, one-step sampling reduces to $\mathbf{z}_0 = \mathbf{z}_1 - \mathbf{u}_\theta(\mathbf{z}_1, 0, 1)$ with $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Improved MeanFlow (iMF) [14] reformulates the objective as a $\mathbf{v}$-loss reparameterized through $\mathbf{u}_\theta$, constructing a *compound* prediction $\mathbf{V}_\theta = \mathbf{u}_\theta + (t-r) \cdot \text{JVP}_{\text{sg}}$ that is regressed against a *guided velocity* $\mathbf{v}_g$. This resolves a variance amplification issue in the original JVP computation. Furthermore, iMF incorporates classifier-free guidance (CFG) directly into training by constructing the guided velocity as:

$$\mathbf{v}_g = \omega\,\mathbf{v}(\mathbf{z}_t \mid \mathbf{c}) + (1-\omega)\,\mathbf{v}(\mathbf{z}_t), \tag{4}$$

where ω is a guidance scale sampled at training time and provided to the network $\mathbf{u}_\theta$ alongside the class label y and time interval $[r, t]$ as global conditioning. This enables flexible adjustment of the guidance strength at inference without retraining. We collectively denote these *standard* conditions (class label, $[r, t]$, and ω) as $\mathbf{c}$ hereafter, distinguishing them from the hierarchy-specific conditions introduced in Sec. 4.2.

x-prediction. Orthogonal to the training objective, the choice of prediction target also matters. Pixel MeanFlow (pMF) [26] introduces a denoised-image field $\hat{\mathbf{x}}(\mathbf{z}_t, r, t) \triangleq \mathbf{z}_t - t \cdot \mathbf{u}(\mathbf{z}_t, r, t)$ and lets the network directly predict the denoised data $\hat{\mathbf{x}}$. The average velocity is recovered via $\mathbf{u}_\theta = \frac{1}{t}(\mathbf{z}_t - \hat{\mathbf{x}}_\theta)$. This parameterization is motivated by the manifold hypothesis adopted from JiT [21]: the denoised output $\hat{\mathbf{x}}$ lies approximately on a low-dimensional data manifold, making it a more tractable regression target than the noisy velocity field [26]. This assumption extends naturally to our setting, where $\hat{\mathbf{x}}$ corresponds to DINOv2 features on a structured representation manifold. We therefore adopt $\mathbf{x}$-prediction combined with $\mathbf{v}$-loss objective as our generative backbone and extend it with hierarchical conditioning in Sec. 4.

Fig. 3: Speed painting illustrates a coarse-to-fine creation process: artists first establish global shape and color blocks before refining local details, yet intermediate stages are already recognizable. (Images courtesy of @yerenhb.)

3.2 Representation Autoencoders

Our framework operates in the feature space of a pretrained visual understanding encoder DINOv2 [30]. This choice is motivated by two properties. **First**, the *representation alignment hypothesis* [45] posits that such feature spaces are geometrically well-organized: semantic factors of variation (e.g., object identity, part structure, spatial layout) are well-separated, so that simple unsupervised clustering over DINOv2 tokens reliably recovers object-part-subpart decompositions without any supervised annotations. We leverage this directly to construct the teacher hierarchies that guide our trajectory design (Sec. 4.1). **Second**, Representation Autoencoders (RAE) [47] train a decoder that reconstructs images directly from DINOv2 features, making any point in this space visually decodable. Because our generation proceeds as a sequence of intermediate refinements in DINOv2 space, the RAE decoder renders every intermediate level as a visual output, enabling inspection and editing before generating the next stage.

4 Method

Artistic training and generative modeling. Human visual artists typically adopt a coarse-to-fine workflow, first establishing global structure and dominant color relationships before refining local details, as illustrated in Fig. 3. Across artistic training practices, beginners are repeatedly advised to avoid *chasing details* and instead focus on structural abstraction and global coherence. The recurrence of this principle across instructors and contexts suggests that it reflects a stable cognitive regularity: global organization precedes and constrains local articulation. Motivated by this observation, we incorporate this structural prior into our model design, translating the coarse-to-fine principle into a hierarchical generation framework where global consistency guides detail refinement.

4.1 Build Teacher Hierarchy

To enable hierarchical generation with intermediate control, we first construct a semantic hierarchy over latent tokens, which serves as a teacher signal for

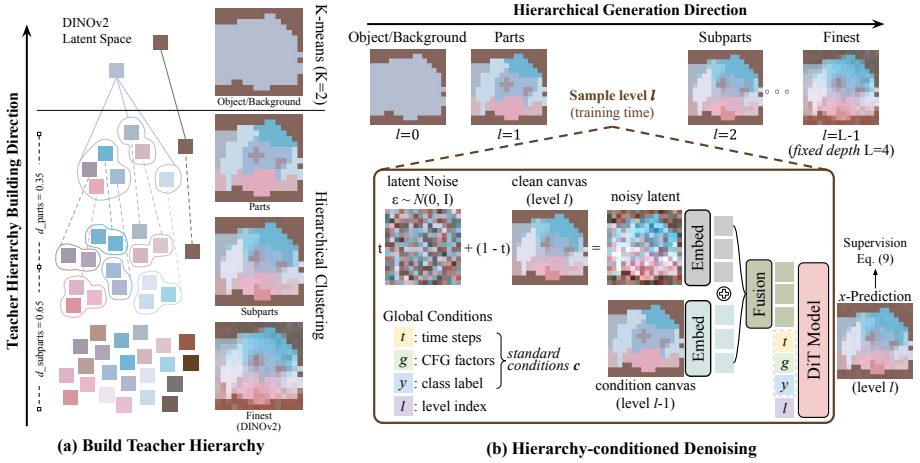


Fig. 4: Trajectory Forcing pipeline. Given an input image, we extract DINOv2 features and construct a teacher hierarchy via unsupervised clustering, producing level canvases from fine (original features, $l = L - 1$) to coarse (object/background, $l = 0$). A single shared network is trained across all levels: at each training step a level l is sampled, and the network denoises the current-level canvas $\mathbf{z}^{(l)}(t)$ conditioned on the previous-level canvas $\mathbf{z}^{(l-1)}$. At inference, the direction reverses: generation proceeds sequentially from coarse to fine. $\oplus$ denotes channel-wise concatenation.

training. Rather than assuming access to human-annotated part hierarchies, we derive this structure directly from pretrained visual representations, leveraging their inherent semantic organization, as illustrated in Fig. 4(a).

Given an input image, we extract dense latent features $\mathbf{z} = \{\mathbf{z}_i\}_{i=1}^{N}$ with $\mathbf{z}_i \in \mathbb{R}^C$, where $N{=}H{\times}W$ is the number of spatial tokens. We assign each token a hierarchical cluster index at **three granularity levels**: object/background, parts, and subparts, via unsupervised clustering, forming a fixed-depth hierarchy shared across all samples.

(i) Object/background: At the coarsest level, we apply K-means with $K{=}2$ in the feature space to obtain two clusters; the cluster whose tokens are on average closer to the image center is designated as object (center prior), and the other as background. This yields a binary partition that serves as the foundation for finer-grained decomposition.

(ii) Part and Subpart: We construct a hierarchy over ***object tokens only*** via agglomerative clustering with a combined *semantic-spatial distance*:

$$\mathcal{D}(i, j) = \cos\langle \mathbf{z}_i, \mathbf{z}_j \rangle + \alpha \, \|\mathbf{p}_i - \mathbf{p}_j\|_2, \tag{5}$$

where $\mathbf{p}_i$ is the normalized spatial coordinate of token i and α controls the spatial regularization strength. This encourages semantically coherent, spatially contiguous clusters. From the resulting dendrogram, we extract a fixed-depth hierarchy by cutting at predefined *distance thresholds*: higher thresholds (0.65) yield finer subparts, lower thresholds (0.35) produce coarser parts.

(iii) Output Hierarchy: The resulting hierarchy assigns each token a tuple of cluster indices (object/background, part, subpart). These indices are stored

as dense maps aligned with the latent grid and serve as region assignments in subsequent stages.

4.2 Hierarchy-conditioned Denoising

Given the teacher hierarchies from Sec. 4.1, we train a hierarchy-conditioned one-step flow model that refines latent features across semantic levels, inducing structured and steerable denoising trajectories.

Level Targets. We construct a set of *level canvases* $\{\mathbf{z}^{(l)}\}_{l=0}^{L-1}$ that serve as denoising targets at each granularity. For levels $l = 0, \ldots, L{-}2$, each token is replaced by the mean feature of its assigned region:

$$\mathbf{z}_i^{(l)} = \boldsymbol{\mu}_{R_i^{(l)}}^{(l)}, \qquad \boldsymbol{\mu}_k^{(l)} = \frac{1}{|R_k^{(l)}|} \sum_{j \in R_k^{(l)}} \mathbf{z}_j, \tag{6}$$

where $R_i^{(l)}$ is the region assignment of token i at level l. At the finest $l = L{-}1$, the canvas is the original feature: $\mathbf{z}^{(L-1)} = \mathbf{z}$. The resulting canvases progress from coarse, piecewise-constant representations to the full-resolution latent, naturally encoding a semantic trajectory from global structure to fine detail.

Level-wise Conditioning. A single shared network is trained across all hierarchy levels. During training, a level index l is sampled uniformly at random for each batch element. Beyond the standard conditions $\mathbf{c}$, the network receives three hierarchy-specific inputs: (1) the noised current-level canvas $\mathbf{z}^{(l)}(t) = (1{-}t)\,\mathbf{z}^{(l)} + t\,\boldsymbol{\epsilon}$, (2) the previous-level canvas $\mathbf{z}^{(l-1)}$ as a spatial condition, and (3) the level index l as a global condition. For level $l{=}0$, the conditioning canvas is set to zero, making the coarsest level unconditional (aside from $\mathbf{c}$).

A natural way to supply the previous-level canvas is *in-context conditioning* [14]: the conditioning tokens are appended to the input token sequence so that the transformer attends over both, but this doubles the sequence length. For efficiency, we instead fuse the two inputs in the channel dimension: the noisy canvas and the previous-level canvas are independently embedded to D dimensions, concatenated to $2D$, and projected back to D via a linear fusion layer, preserving the original sequence length. The level index is encoded through a learned embedding table and injected as additional conditioning tokens alongside class and guidance embeddings.

Total Training Objective.
(i) Flow loss: We adopt $\mathbf{x}$-prediction combined with $\mathbf{v}$-loss [26]. The network predicts denoised latents $\hat{\mathbf{x}}_\theta(\mathbf{z}^{(l)}(t),\, l,\, \mathbf{z}^{(l-1)};\, \mathbf{c})$, from which the average velocity is recovered as $\mathbf{u}_\theta = (\mathbf{z}^{(l)}(t) - \hat{\mathbf{x}}_\theta)/t$. The compound prediction $\mathbf{V}_\theta = \mathbf{u}_\theta + (t{-}r)\cdot \mathrm{JVP}_{\mathrm{sg}}$ is constructed and regressed against the guided velocity $\mathbf{v}_g$:

$$\mathcal{L}_{\mathrm{flow}} = \mathbb{E}_{t,r,l}\left[\, \|\mathbf{V}_\theta - \mathbf{v}_g\|^2 \,\right]. \tag{7}$$

(ii) Structural loss: To encourage the prediction to respect region structure, we penalize the deviation of each predicted token from the ground-truth mean of its assigned region. Let $\boldsymbol{\mu}_k^{(l)}$ denote the target region mean as in Eq. (6). For each token i in region k, we compute the squared cosine distance between the predicted token $\hat{\mathbf{x}}_{\theta,i}$ and its region's target mean, averaged over all valid tokens

and regions:

$$\mathcal{L}_{\text{struct}} = \frac{1}{K_l} \sum_{k=1}^{K_l} \frac{1}{|R_k^{(l)}|} \sum_{i \in R_k^{(l)}} \left(1 - \cos\left\langle \hat{\mathbf{x}}_{\theta,i},\, \boldsymbol{\mu}_k^{(l)} \right\rangle\right)^2, \tag{8}$$

where K_l is the number of valid regions. This loss is applied to $l \in \{0, \ldots, L-2\}$ and disabled at the finest level, where no region structure is imposed.

Overall objective. The full training loss is:

$$\mathcal{L} = \mathcal{L}_{\text{flow}} + \lambda\, \mathcal{L}_{\text{struct}}, \tag{9}$$

where λ controls the structural loss weight.

4.3 Hierarchical Sampling

At inference, generation proceeds sequentially through the hierarchy from $l = 0$ to $L-1$. At the coarsest level, the model generates from pure noise with zero spatial conditioning: $\hat{\mathbf{z}}^{(0)} := \hat{\mathbf{x}}_\theta(\boldsymbol{\epsilon},\, l{=}0,\, \mathbf{0};\, \mathbf{c})$. For each subsequent level $l > 0$, fresh noise is drawn and the model generates conditioned on the output of the previous level:

$$\hat{\mathbf{z}}^{(l)} := \hat{\mathbf{x}}_\theta(\boldsymbol{\epsilon}',\, l,\, \hat{\mathbf{z}}^{(l-1)};\, \mathbf{c}). \tag{10}$$

Each level requires a single network function evaluation (NFE), yielding a total cost of L-NFE for the complete hierarchy. Crucially, every intermediate output $\hat{\mathbf{z}}^{(l)}$ is decodable into a visual image via the RAE decoder, providing a meaningful preview at each generation stage.

4.4 Interactive Generation and Editing

Because generation is decomposed into discrete, interpretable levels, users can inspect, modify, and re-generate at any point along the trajectory.

Editing workflow. Given a fully generated trajectory $\{\hat{\mathbf{z}}^{(l)}\}_{l=0}^{L-1}$, a user inspects the decoded output at level l^* and produces an edited canvas $\tilde{\mathbf{z}}^{(l^*)}$. Generation then resumes from this edit: levels $l > l^*$ are re-generated conditioned on the modified output, while all coarser levels $l < l^*$ remain unchanged. This workflow supports two complementary editing operations:

(i) Feature editing: The user replaces the mean feature of a selected region in $\hat{\mathbf{z}}^{(l^*)}$ with a feature sourced from another region or image (e.g., swapping one part's semantics with another), then propagates forward through subsequent levels. Because semantically similar content maps to nearby features in DINOv2 space, this transfers the semantic identity of the source region to the target.

(ii) Shape editing: The user modifies the spatial extent of a region by reassigning tokens at the boundary between adjacent regions in $\hat{\mathbf{z}}^{(l^*)}$, then propagates forward. This changes *where* a region is without altering its feature content, for example, enlarging or shrinking a part, or reshaping its contour.

A key property underlying both operations is editing *scope control*. Since generation is Markov (each level conditioned only on the preceding one), an edit at level l^* propagates through levels $l > l^*$ but leaves coarser levels untouched. This gives users predictable control over editing scope: coarser edits cascade through more stages and have broader semantic impact, while finer edits remain spatially localized. These editing operations motivate the trajectory-aware evaluation metrics introduced in Sec. 5.

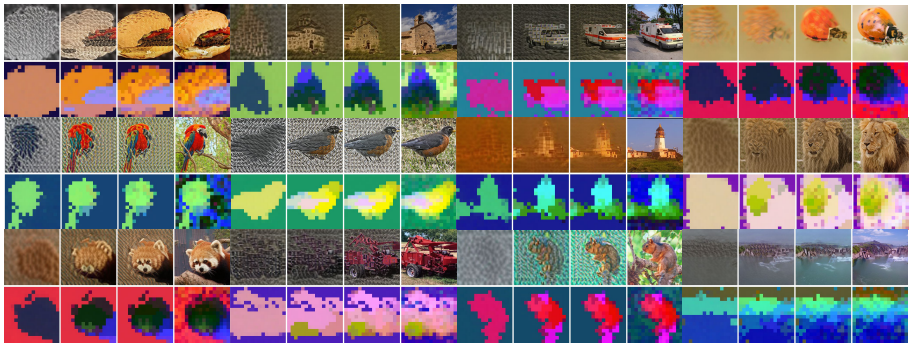


Fig. 5: Decoded samples with PCA colored latent stages.

5 ImageNet Experiments

We evaluate on ImageNet [6] at 256×256 resolution. Following the hierarchical sampling of Sec. 4.3, each image is produced in L=4 one-step denoising stages in DINOv2 [30] feature space, with every intermediate output decodable via a pre-trained ViT-XL decoder [47]. We report FID [16] and IS [34] on 50k samples.

Our backbone is a DiT [31] transformer shared across all levels, with patch size 16×16 (denoted TF/16). By default, we train TF with structural loss weight λ=1 and the Muon optimizer [18] at a constant learning rate of $1e-2$. Implementation details, full hyperparameters, scalability across model sizes, and λ ablation are provided in the supplementary material.

5.1 System-level Comparison

We compare with prior methods in Tab. 1. Since work on hierarchical latent generation is scarce, we include representative pixel-space as well as latent-space baselines, spanning both multi-step and one-step diffusion/flow approaches. Unless otherwise noted, we report results for models trained from scratch.

At 80 training epochs, TF shows rapid convergence relative to baselines that typically require several hundred epochs. We note that these are early-training results; the primary contribution of TF is not FID optimization but trajectory-level controllability, which no other method in the table provides.

Relation to prior work. The most related prior work is NVG [41], which also performs hierarchical generation; we discuss conceptual differences in Sec. 2. From a practical standpoint, two differences are worth highlighting: **(i)** NVG requires training a custom multi-granularity VQ-VAE to support its resolution-based hierarchy, whereas TF operates directly in pretrained DINOv2 features without a task-specific tokenizer; **(ii)** NVG's structure generation relies on a multi-step rectified flow model at each stage (9 content steps + 7×25 Euler steps = 184 NFEs); moreover, its discrete nature makes it difficult to leverage recent one-step advances [7, 14, 26], whereas TF naturally integrates one-step matching and produces each level in a single evaluation (L=4 NFEs total).

5.2 Evaluation beyond FID

Standard metrics such as FID evaluate distributional quality, but do not capture the controllability and structural faithfulness central to TF. To measure

Table 1: Comparison on ImageNet 256×256. FID and IS are computed on 50k samples with CFG where applicable (×2 on NFE indicates CFG doubles the cost). We compare against baselines of comparable model size and, when applicable, similar training epochs; extended comparisons can be found in in the supplementary.

ImgNet 256×256	NFE	Space	Params	Epochs	FID(↓)	IS(↑)
Multi-step pixel diffusion/flow						
JiT-B/16 (Heun) [21]	100 × 2	pixel	131M	200	4.37	275.1
JiT-L/16 (Heun)	100 × 2	pixel	459M	200	2.36	298.5
DeCo-XL/16 [28]	100 × 2	pixel	682M	320	1.90	303.0
LF-ViT/L (Heun) [2]	50 × 2	pixel+DINOv2	465M	200	2.48	–
Multi-step latent diffusion/flow						
DiT-B/2 [31]	250	SD-VAE	130M	80	43.47	278.2
DiT-XL/2	250	SD-VAE	675M	1400	9.62	121.5
DiT-XL/2 (cfg=1.50)	250 × 2	SD-VAE	675M	1400	2.27	278.2
SiT-B/2 [27]+REPA [45]	250	SD-VAE	130M	80	24.40	–
SiT-XL/2	250	SD-VAE	675M	1400	8.61	131.7
SiT-XL/2 (cfg=1.50)	250 × 2	SD-VAE	675M	1400	2.06	270.3
RAE+DiT^{DH}-XL/2 [47]	50 × 2	DINOv2	839M	800	1.13	262.6
Autoregressive latent diffusion/flow						
VAR-d16 [38]	250	VQ-VAE	310M	200	3.30	274.4
VAR-d20	250	VQ-VAE	600M	250	2.57	302.64
NVG-d16 [41]	184*	VQ-VAE	255M	200	3.03	279.2
NVG-d20	184*	VQ-VAE	497M	250	2.44	310.4
1-NFE diffusion/flow						
pMF-B/16 [26]	1	pixel	119M	320	3.12	254.6
pMF-L/16	1	pixel	411M	320	2.52	262.6
EPG-B/16 [20]	1	pixel	229M	240	25.10	–
EPG-L/16 [20]	1	pixel	540M	560	8.82	–
iCT-XL/2 [36]	1	SD-VAE	675M	240	34.24†	–
Shortcut-XL/2 [9]	1	SD-VAE	676M	240	10.60†	102.7
IMM-XL/2 [48]	1×2	SD-VAE	676M	240	7.77†	–
MeanFlow-B/4 [13]	1	SD-VAE	131M	80	15.53	–
iMF-M/2 [14]	1	SD-VAE	174M	640	2.27	–
iMF-L/2 [14]	1	SD-VAE	409M	640	1.86	–
TF-B/16 (ours)	4×1	DINOv2	177M	80	7.93	221.4
TF-B/16+FD-loss‡ (ours)	4×1	DINOv2	515M	80	2.52	245.6
TF-L/16 (ours)	4×1	DINOv2	177M	80	6.38	248.1
TF-L/16+FD-loss‡ (ours)	4×1	DINOv2	515M	80	2.02	259.3
TF-H/16 (ours)	4×1	DINOv2	1.1B	80	5.97	256.8
TF-H/16+FD-loss‡ (ours)	4×1	DINOv2	1.1B	80	1.98	261.6

*NVG requires 184 NFEs total; see text Sec. 5.1 for breakdown.

†Results reported from the original MeanFlow paper (trained from scratch).

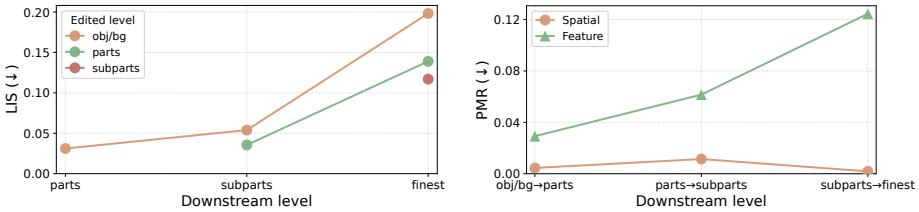
‡Inception-space FD-loss [44] post-training; see text App. A.1 for details.

these properties, we introduce trajectory-aware metrics along two complementary axes: (i) spatial locality of edits and (ii) structural coherence across levels.

5.2.1 Local Invariance under Manipulation. We evaluate whether editing at level ℓ^* introduces unintended changes to regions that are not explicitly modified. Let $\hat{\mathbf{z}}^{(\ell)}$ denote the generated latent at level ℓ before manipulation and $\hat{\mathbf{z}}'^{(\ell)}$ the corresponding latent after re-generation from the edited canvas. Given a spatial mask $\mathbf{M} \in \{0,1\}^{H\times W}$ indicating edited tokens (M_{ij}=1), we define the **Latent Invariance Score (LIS)** as the average cosine distance over unedited tokens:

$$f_{\text{lis}}^{(\ell)} = \frac{1}{|\Omega_{\text{unedit}}|} \sum_{(i,j)\in\Omega_{\text{unedit}}} \left(1 - \cos\left\langle \hat{\mathbf{z}}_{i,j}^{(\ell)}, \hat{\mathbf{z}}_{i,j}'^{(\ell)} \right\rangle\right), \tag{11}$$

where $\Omega_{\text{unedit}} = \{(i,j) \mid M_{ij}=0\}$. Lower scores indicate stronger local invariance. We evaluate LIS at each downstream level $\ell \geq \ell^*$ to study how edits propagate



(a) Local Invariance Score (LIS) **(b) Structural Consistency (PMR)**

Fig. 6: Trajectory-aware evaluation. (a) LIS shows coarser edits propagate more strongly, while finer edits remain localized. (b) Both spatial and feature PMR remain low, indicating hierarchical consistency across levels. We evaluate on TF alone as no baseline produces semantic-region intermediates (NVG uses binary splits over discrete tokens without region structure).

through subsequent generation stages. Fig. 6(a) shows a clear scale-dependent behavior: coarser edits produce larger downstream deviations, whereas finer edits remain more localized. Object/background edits yield the highest LIS at the final level, followed by parts and subparts edits. This confirms that TF exposes a controllable hierarchy where the spatial extent of changes can be modulated by the editing level.

5.2.2 Structural Consistency across Levels. Beyond editing invariance, we evaluate structural coherence between consecutive levels: each child-level region should be spatially contained within a single parent region; a child straddling multiple parents signals structural drift.

To formalize this, given generated outputs at consecutive levels ℓ and $\ell{+}1$, we cluster each into regions using the procedure of Sec. 4.1. At level ℓ, we obtain K_ℓ parent regions with mean-feature centers $\{\boldsymbol{\mu}_i^{(\ell)}\}_{i=1}^{K_\ell}$ and per-token assignments $a^{(\ell)}(n) \in \{1, \ldots, K_\ell\}$. At level $\ell{+}1$, we obtain $K_{\ell+1}$ child regions $\{C_j\}_{j=1}^{K_{\ell+1}}$.

Since a child region may partially overlap multiple parent regions near boundaries, we assign each child a parent by spatial majority:

$$\text{parent}(j) = \arg\max_i \left| \{n \in C_j : a^{(\ell)}(n) = i\} \right|.$$

This induces a token-level expected parent: every token n in child region j inherits parent(j) as its expected parent. We define two complementary token-level **Parent Misrouting Rates (PMR)**:

(i) Spatial PMR. The fraction of tokens whose parent-level cluster assignment disagrees with the majority-voted parent of their child region:

$$f_{\text{pmr-s}}^{(\ell)} = \mathbb{P}_n \left[a^{(\ell)}(n) \neq \text{parent}(j_n) \right], \tag{12}$$

where j_n denotes the child region containing token n. This measures whether child regions are cleanly contained within single parent regions.

(ii) Feature PMR. The fraction of tokens whose feature is closer to an incorrect parent center than to the assigned one:

$$f_{\text{pmr-f}}^{(\ell)} = \mathbb{P}_n \left[\arg\max_i \cos\left\langle \hat{\mathbf{z}}_n^{(\ell+1)}, \boldsymbol{\mu}_i^{(\ell)} \right\rangle \neq \text{parent}(j_n) \right]. \tag{13}$$

This captures whether generation has shifted token features away from their assigned parent's semantic mode.

Lower values indicate stronger structural consistency; both metrics are evaluated at each consecutive level pair. Fig. 6(b) shows that spatial PMR remains near zero across levels, indicating that child regions are largely contained within their parent regions. Feature PMR is slightly higher and increases at deeper levels due to finer semantic partitioning, but remains low overall. These results indicate that TF preserves hierarchical consistency while progressively refining semantic detail.

Edit effectiveness and decoded-space checks. The metrics above verify that edits do not disrupt unedited regions and that generation is structurally coherent. We additionally evaluate whether edits achieve their intended effect in the target region, and verify that latent-level properties translate to pixel space by decoding intermediate outputs via the frozen RAE decoder. Editing examples and decoded-space analysis are provided in the supplementary material.

Why a shared decoder? A natural question is whether a single decoder suffices for all levels, since intermediate canvases differ in distribution from the final latent. We deliberately avoid per-level finetuning for two reasons: first, a shared decoder ensures that decoded images across levels live in a consistent visual space, which is essential for editing: a user must be able to compare the output at level ℓ^* with the final image; second, no ground-truth intermediate images exist, as level canvases are piecewise-constant abstractions in feature space, making per-level supervision unavailable by construction.

6 Discussion and Conclusions

We presented Trajectory Forcing, an approach that treats the generative trajectory as a first-class, user-facing object rather than a hidden computational process. By operating in a semantically structured representation space and combining hierarchical conditioning with one-step flow matching, TF produces a coarse-to-fine sequence of decodable, editable intermediate states in L=4 NFE.

Limitations. Our hierarchy is derived from unsupervised clustering with a fixed depth and a center prior for object/background separation, which may not generalize well to images without a centered dominant object. Because each level is conditioned only on the immediately preceding one, coarser-level information reaches finer levels indirectly through the chain rather than via direct conditioning. Current results are reported at 40/80 training epochs; longer training and scaling analysis are deferred to the supplementary material.

Future work. Natural extensions include replacing the fixed clustering pipeline with richer hierarchy sources (such as model segmentations [3]), adaptive hierarchy depth per image, and extending TF to text-conditional generation.

Acknowledgements

Merve Kocabas is supported by the Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA) through the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the German Federal Ministry of Education and Research. Gege Gao acknowledges the EuroHPC Joint Undertaking for awarding access to computational resources under project ID EHPC-AIF-2026PG01-147 on the Discoverer+ GPU partition hosted by Sofia Tech Park. Andreas Geiger is a member of the Machine Learning Cluster of Excellence, EXC 2064/1, project number 390727645. The authors gratefully acknowledge the ML Cloud and the Tübingen AI Center for providing the computational resources and facilities that supported this work.

References

1. Atanov, A., Allardice, J., Bachmann, R., Kar, O.F., Fu, P., Griffiths, D., Hjelm, D., Dehghan, A., Zamir, A.: VideoFlexTok: Flexible-length coarse-to-fine video tokenization. In: Proc. of the International Conf. on Machine learning (ICML) (2026)
2. Baade, A., Chan, E.R., Sargent, K., Chen, C., Johnson, J., Adeli, E., Fei-Fei, L.: Latent forcing: Reordering the diffusion trajectory for pixel-space image generation. arXiv preprint arXiv:2602.11401 (2026)
3. Carion, N., Gustafson, L., Hu, Y.T., Debnath, S., Hu, R., Suris, D., Ryali, C., Alwala, K.V., Khedr, H., Huang, A., et al.: Sam 3: Segment anything with concepts. arXiv preprint arXiv:2511.16719 (2025)
4. Chen, B., Monsó, D.M., Du, Y., Simchowitz, M., Tedrake, R., Sitzmann, V.: Diffusion forcing: Next-token prediction meets full-sequence diffusion. In: Advances in Neural Information Processing Systems (NeurIPS) (2025)
5. Chen, S., Ge, C., Zhang, S., Sun, P., Luo, P.: Pixelflow: Pixel-space generative models with flow. arXiv preprint arXiv:2504.07963 (2025)
6. Deng, Jia, Dong, Wei, Socher, Richard, Li, Li-Jia, Li, Kai, Fei-Fei, Li: Imagenet: A large-scale hierarchical image database. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2009)
7. Deng, M., Li, H., Li, T., Du, Y., He, K.: Generative modeling via drifting. arXiv preprint arXiv:2602.04770 (2026)
8. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2021)
9. Frans, K., Hafner, D., Levine, S., Abbeel, P.: One step diffusion via shortcut models. In: Proc. of the International Conf. on Learning Representations (ICLR) (2025)
10. Gao, G., Huang, H., Fu, C., He, R.: Causal representation learning for context-aware face transfer. arXiv preprint arXiv:2110.01571 (2021)
11. Gao, G., Liu, W., Chen, A., Geiger, A., Schölkopf, B.: Graphdreamer: Compositional 3d scene synthesis from scene graphs. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2024)
12. Gao, G., Schölkopf, B., Geiger, A.: Slots, transitions, loops: Learning composable world models for arc. arXiv preprint arXiv:2606.12316 (2026)
13. Geng, Z., Deng, M., Bai, X., Kolter, J.Z., He, K.: Mean flows for one-step generative modeling. In: Advances in Neural Information Processing Systems (NeurIPS) (2025)

14. Geng, Z., Lu, Y., Wu, Z., Shechtman, E., Kolter, J.Z., He, K.: Improved mean flows: On the challenges of fastforward generative models. arXiv preprint arXiv:2512.02012 (2025)
15. Gupta, S., Jalal, A., Parulekar, A., Price, E., Xun, Z.: Diffusion posterior sampling is computationally intractable. In: Proc. of the International Conf. on Machine learning (ICML) (2024)
16. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems (NeurIPS) (2017)
17. Hu, Z., Lai, C.H., Wu, G., Mitsufuji, Y., Ermon, S.: Meanflow transformers with representation autoencoders. arXiv preprint arXiv:2511.13019 (2025)
18. Jordan, Keller, Jin, Yuchen, Boza, Vlado, Jiacheng, You, Cecista, Franz, Newhouse, Laker, Bernstein, Jeremy: Muon: An optimizer for hidden layers in neural networks (2024), https://kellerjordan.github.io/posts/muon
19. Kim, D., Lai, C.H., Liao, W.H., Murata, N., Takida, Y., Uesaka, T., He, Y., Mitsufuji, Y., Ermon, S.: Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In: Proc. of the International Conf. on Learning Representations (ICLR) (2024)
20. Lei, J., Liu, K., Berner, J., HoiM, Y., Zheng, H., Wu, J., Chu, X.: There is no vae: Advancing end-to-end pixel-space generative modeling via self-supervised pre-training. In: Proc. of the International Conf. on Learning Representations (ICLR) (2026)
21. Li, T., He, K.: Back to basics: Let denoising generative models denoise. arXiv preprint arXiv:2511.13720 (2026)
22. Li, T., Tian, Y., Li, H., Deng, M., He, K.: Autoregressive image generation without vector quantization. In: Advances in Neural Information Processing Systems (NeurIPS) (2024)
23. Li, Y., Bornschein, J., Chen, T.: Denoising autoregressive representation learning. In: Proc. of the International Conf. on Machine learning (ICML) (2024)
24. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: Learning to generate and transfer data with rectified flow. arXiv preprint arXiv:2209.03003 (2022)
25. Lu, C., Song, Y.: Simplifying, stabilizing and scaling continuous-time consistency models. In: Proc. of the International Conf. on Learning Representations (ICLR) (2025)
26. Lu, Y., Lu, S., Sun, Q., Zhao, H., Jiang, Z., Wang, X., Li, T., Geng, Z., He, K.: One-step latent-free image generation with pixel mean flows. arXiv preprint arXiv:2601.22158 (2026)
27. Ma, N., Goldstein, M., Albergo, M.S., Boffi, N.M., Vanden-Eijnden, E., Xie, S.: Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In: European Conference on Computer Vision. pp. 23–40. Springer (2024)
28. Ma, Z., Wei, L., Wang, S., Zhang, S., Tian, Q.: Deco: Frequency-decoupled pixel diffusion for end-to-end image generation. arXiv preprint arXiv:2511.19365 (2025)
29. Meng, C., Rombach, R., Gao, R., Kingma, D.P., Ermon, S., Ho, J., Salimans, T.: On distillation of guided diffusion models. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2023)
30. Oquab, M., Darcet, T., Moutakanni, T., Vo, H.V., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
31. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV) (2023)

32. Ren, S., Yu, Q., He, J., Shen, X., Yuille, A., Chen, L.C.: Flowar: Scale-wise autoregressive image generation meets flow matching. arXiv preprint arXiv:2412.15205 (2024)
33. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2022)
34. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in Neural Information Processing Systems (NeurIPS) (2016)
35. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. In: Proc. of the International Conf. on Learning Representations (ICLR) (2022)
36. Song, Y., Dhariwal, P.: Improved techniques for training consistency models. In: Proc. of the International Conf. on Learning Representations (ICLR) (2024)
37. Song, Y., Dhariwal, P., Chen, M., Sutskever, I.: Consistency models. In: Proc. of the International Conf. on Machine learning (ICML) (2023)
38. Tian, K., Jiang, Y., Yuan, Z., PENG, B., Wang, L.: Visual autoregressive modeling: Scalable image generation via next-scale prediction. In: Advances in Neural Information Processing Systems (NeurIPS) (2024)
39. Venkataramanan, S., Pariza, V., Salehi, M., Knobel, L., Gidaris, S., Ramzi, E., Bursuc, A., Asano, Y.M.: Franca: Nested matryoshka clustering for scalable visual representation learning. arXiv preprint arXiv:2507.14137 (2025)
40. Wang, S., Gao, Z., Zhu, C., Huang, W., Wang, L.: Pixnerd: Pixel neural field diffusion. In: Proc. of the International Conf. on Learning Representations (ICLR) (2026)
41. Wang, Y., Wang, Z., Wu, Z., Tao, Q., Liao, K., Loy, C.C.: Next visual granularity generation. In: Proc. of the International Conf. on Learning Representations (ICLR) (2026)
42. Wang, Z., Zhang, Y., Yue, X., Yue, X., Li, Y., Ouyang, W., Bai, L.: Transition models: Rethinking the generative learning objective. arXiv preprint arXiv:2509.04394 (2025)
43. Wei, C., Mangalam, K., Huang, P.Y., Li, Y., Fan, H., Xu, H., Wang, H., Xie, C., Yuille, A., Feichtenhofer, C.: Diffusion models as masked autoencoders. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV) (2023)
44. Yang, J., Geng, Z., Ju, X., Tian, Y., Wang, Y.: Representation fréchet loss for visual generation. arXiv preprint arXiv:2604.28190 (2026)
45. Yu, S., Kwak, S., Jang, H., Jeong, J., Huang, J., Shin, J., Xie, S.: Representation alignment for generation: Training diffusion transformers is easier than you think. In: Proc. of the International Conf. on Learning Representations (ICLR) (2025)
46. Zhang, H., Siarohin, A., Menapace, W., Vasilkovsky, M., Tulyakov, S., Qu, Q., Skorokhodov, I.: Alphaflow: Understanding and improving meanflow models. arXiv preprint arXiv:2510.20771 (2025)
47. Zheng, B., Ma, N., Tong, S., Xie, S.: Diffusion transformers with representation autoencoders. In: Proc. of the International Conf. on Learning Representations (ICLR) (2026)
48. Zhou, L., Ermon, S., Song, J.: Inductive moment matching. In: Proc. of the International Conf. on Machine learning (ICML) (2025)