FROM PIXELS, WITHOUT PRE-TRAINING: JOINT GENERATIVE AND SELF-SUPERVISED REPRESENTATION LEARNING IN ONE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Strong image generation models today are conditioned on class labels, aligned to frozen pretrained encoders, or built on separately trained autoencoders or tokenizers. While effective, the cost is that generation becomes contingent on supervision and pretraining: labels must be annotated, and encoders, autoencoders, and tokenizers must be pretrained for the target domain. This paper studies **joint generative and self-supervised representation learning in one model**, such that generation is self-conditioned without any labels or pretrained models. This is challenging since the two objectives are mismatched: contrastive learning consumes clean augmented views and keeps coarse, invariant semantics, while flow matching consumes noisy images and must preserve the fine detail and spatial layout that contrastive learning discards. We propose a framework Self-conditioned Generation on Self-supervised representation (SCION), whose core is a single pixel-space encoder conditioned on both the diffusion timestep and a conditioning embedding. The encoder serves two roles that differ in what that embedding is. For representation learning, the conditioning embedding is one learned global vector shared by all images, and the encoder's `[CLS]` token yields the semantic embedding trained by the contrastive loss. For generative training, the conditioning embedding is the `[CLS]` embedding of the image itself, and the encoder's patch tokens pass through a decoder to predict the image. To sample without a reference image at inference, we also learn a prior over the embedding in the same run. With gradient-norm balancing and stop-gradient tricks, all losses are jointly optimized in a single run. The resulting method is fully self-supervised, with no pretrained model and no labels. On ImageNet 256×256, with the same JiT-B/16 recipe and without representation guidance, SCION reaches 8.92 FID, surpassing the class-unconditional baselines iREPA, which aligns to a pretrained DINOv2 (46.44), and RCG, which conditions on it (14.27). Under the JiT-L/16 recipe, SCION achieves the best pixel-space class-unconditional FID of 5.89 and 3.47 without and with representation guidance, outperforming RCG with the ADM recipe (6.24).

1 INTRODUCTION

Diffusion and flow matching models have achieved remarkable success in image generation (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023; Liu et al., 2023). Yet many strong systems rely on class labels for semantic conditioning (Dhariwal & Nichol, 2021; Li & He, 2026), frozen pretrained encoders for conditioning or representation alignment (Li et al., 2024a; Yu et al., 2025; Singh et al., 2026), or separately trained autoencoders for latent-space generation (Rombach et al., 2022). These dependencies add annotation or pretraining costs and complicate extension to domains where suitable labels or pretrained models are unavailable. A central challenge is therefore to enable *semantically self-conditioned generation*, where the semantic representations that condition generation are learned within the diffusion model itself, jointly with the generative process.

We pursue semantic self-conditioned generation under four "S" requirements: *self-contained*, without pretrained components, including autoencoders and tokenizers; *self-supervised*, without label supervision anywhere in the pipeline, including that inherited through pretrained components; *single-model*, with representations produced by the generator's own network; and *single-stage*, with all components

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.

learned jointly from scratch. Existing methods (cf. Table 1) address different aspects of this goal but satisfy these requirements only partially (Li & He, 2026; Yu et al., 2025; Li et al., 2024a; 2023; Preechakul et al., 2022; Hudson et al., 2024; Wang & He, 2025). These comparisons highlight the remaining challenge: achieving semantic self-conditioning under all four "S" requirements, with useful representations and unconditional synthesis learned together within one model.

A natural approach to semantically self-conditioned generation is to learn the conditioning representations through contrastive self-supervised learning, which extracts semantics directly from unlabeled images (Chen et al., 2020a; He et al., 2020; Yuan et al., 2022). However, integrating self-supervised representation learning with flow matching in one network requires reconciling their conflicting input and output requirements. Contrastive learning operates on clean augmented views and encourages compact semantic embeddings that suppress variations in appearance and spatial layout. Flow matching instead operates on noisy images and requires dense predictions that recover fine visual detail and spatial structure. The shared network must therefore accommodate both clean and noisy inputs while supporting semantic invariance without discarding information needed for synthesis. Joint training further couples the generator to an evolving conditioning representation, and sampling requires obtaining conditioning embeddings without a reference image.

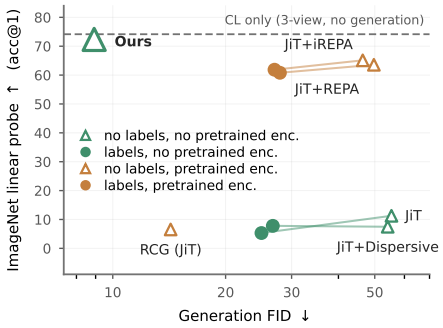

Figure 1: **Generation and Representation capabilities** of different models trained on ImageNet 256×256 with 400 epochs. FID under the ADM evaluation protocol. A segment joins a method's class-conditional and class-unconditional variants, so its length is what the class label buys.

To address these challenges, we propose **S**elf-**C**onditioned Generation on Self-Supervised Representat**ion** (SCION), a framework for semantically self-conditioned generation through joint generative and self-supervised representation learning. At its core is a single pixel-space transformer encoder conditioned on both a timestep and a conditioning embedding. Our key idea is to use this encoder in two conditioning modes with different readouts. For representation learning, the encoder processes clean augmented views using a learned global conditioning vector shared across all images, and a contrastive objective trains its `[CLS]` embedding. For generative learning, the same encoder processes noisy images conditioned on the clean image's own `[CLS]` embedding, and a decoder maps its patch tokens to a clean-image prediction under a flow matching objective. This design keeps contrastive inputs clean while using spatial features for generation, allowing both objectives to train the shared encoder.

To coordinate joint optimization, we employ gradient-norm balancing and stop-gradient operations. We also train a prior over the learned embeddings in the same run, enabling generation without reference images. Randomly withholding the conditioning embedding during training also supports classifier-free guidance (Ho & Salimans, 2021), with guidance supplied by learned representations rather than class labels. All components are trained jointly from scratch, without labels, pretrained teachers, latent autoencoders, or tokenizers.

We evaluate on ImageNet at 256×256 against flow-based baselines matched on the JiT-B/16 recipe. SCION reaches 8.92 FID, improving by $6.1\times$ over the strongest self-contained baseline (54.14), and outperforming methods that import pretrained semantics, either aligning with them (iREPA, 46.44) or conditioning on them (RCG at matched training budget, 14.27). Self-representation guidance further reduces SCION to 4.20 FID, outperforming representation-guided RCG (5.01) without importing a pretrained encoder. At larger scale, SCION (JiT-L/16) reaches 5.89 unguided FID, improving on the 6.24 reported by pixel-space RCG with an ADM training recipe. As Figure 1 shows, SCION's learned representation also remains competitive with contrastive-only training.

2 Preliminaries

Let $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$ be a dataset of images sampled from an unknown data distribution p_{data}, which we want to model. For contrastive learning, we let $A_a, A_b \sim \mathcal{A}$ denote two random augmentations from a predefined augmentation policy $\mathcal{A}$ (e.g., random crop, horizontal flip). For flow matching, we

Table 1: **Comparison with related works under the four S requirements.** Extended discussion on related works is presented in Appendix A. **Self-contained** means no pretrained component; **self-supervised** means no label supervision anywhere in the pipeline; **single-model** means that the representation is produced by the generator's own network; and **single-stage** means that all components are learned jointly from scratch. Capabilities report a learned and evaluated **Repr**esentation and unconditional **Synth**esis with no source image to encode. All pipelines are as published, except REPA and Dispersive, which we combine with the pixel-space JiT recipe.

	four S requirements				*capabilities*	
Method	Self-cont.	Self-sup.	Single-mod.	Single-stage	Repr.	Synth.
Pixel-space generation						
JiT (Li & He, 2026)	✓	✗	–	✓	✗	✓
JiT + REPA (Yu et al., 2025)	✗	✗	✗	✗	✗	✓
JiT + Dispersive (Wang & He, 2025)	✓	✗	✓	✓	✗	✓
Semantic conditions						
RCG (ADM) (Li et al., 2024a)	✗	✓	✗	✗	✗	✓
Joint representation and generation						
MAGE (Li et al., 2023)	✗	✗	✓	✗	✓	✓
DiffAE (Preechakul et al., 2022)	✓	✓	✗	✗	✓	✓
SODA (Hudson et al., 2024)	✓	✓	✗	✓	✓	✗
SCION (ours)	✓	✓	✓	✓	✓	✓

sample a timestep $t \in [0, 1]$ from a logit-normal distribution $\mathrm{logit}(t) \sim \mathcal{N}(\mu_t, \sigma_t^2)$, and a noise vector $\boldsymbol{\epsilon} \sim p_{\mathrm{noise}}$ where $p_{\mathrm{noise}} = \mathcal{N}(0, \boldsymbol{I})$ is a standard Gaussian distribution. Let $\|\cdot\|$ denote the Euclidean norm and let $\mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v}/(\|\boldsymbol{u}\|\,\|\boldsymbol{v}\|)$ denote the cosine similarity between two vectors.

2.1 Flow Matching

Flow matching (Lipman et al., 2023; Liu et al., 2023; Albergo & Vanden-Eijnden, 2023) is a generative modeling framework that learns a continuous-time vector field whose integral curves transport a noise distribution p_{noise} to the data distribution p_{data}. Its trajectories follow the ordinary differential equation (ODE) $d\boldsymbol{x}_t/dt = \boldsymbol{v}_t(\boldsymbol{x}_t)$ started at $\boldsymbol{x}_0 = \boldsymbol{\epsilon} \sim p_{\mathrm{noise}}$, so that $\boldsymbol{x}_1 \sim p_{\mathrm{data}}$: the velocity field is the time derivative of this *marginal* path, and solving the ODE is what generates samples.

The marginal field is defined from a *conditional* path between a noise sample and a data sample, which we denote by $\widetilde{\boldsymbol{x}}_t$. Most works choose the conditional path according to the linear interpolant:

$$\widetilde{\boldsymbol{x}}_t = t\,\boldsymbol{x} + (1-t)\,\boldsymbol{\epsilon}, \qquad \boldsymbol{x} \sim p_{\mathrm{data}}, \quad \boldsymbol{\epsilon} \sim p_{\mathrm{noise}}.$$

Differentiating it gives the conditional velocity $\boldsymbol{v}_t(\widetilde{\boldsymbol{x}}_t \mid \boldsymbol{x}, \boldsymbol{\epsilon}) = d\widetilde{\boldsymbol{x}}_t/dt = \boldsymbol{x} - \boldsymbol{\epsilon}$, which depends on the endpoints rather than on the current state alone. Following Li & He (2026), instead of directly parameterizing the velocity field $\boldsymbol{v}_t(\widetilde{\boldsymbol{x}}_t \mid \boldsymbol{x}, \boldsymbol{\epsilon})$, we use the fact that $\boldsymbol{v}_t(\widetilde{\boldsymbol{x}}_t \mid \boldsymbol{x}, \boldsymbol{\epsilon}) = (\boldsymbol{x} - \widetilde{\boldsymbol{x}}_t)/(1-t)$ and predict the clean image $\boldsymbol{x}$ instead. Thus, we parameterize a neural network $F_\theta(\widetilde{\boldsymbol{x}}_t, t)$ that predicts the clean image $\boldsymbol{x}$ from the noised input $\widetilde{\boldsymbol{x}}_t$ and time t. Then, the Conditional Flow Matching (CFM) objective is given by:

$$\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t, \boldsymbol{\epsilon} \sim p_{\mathrm{noise}}, \boldsymbol{x} \sim p_{\mathrm{data}}} \left[\left\| \frac{F_\theta(\widetilde{\boldsymbol{x}}_t, t) - \widetilde{\boldsymbol{x}}_t}{1-t} - \frac{\boldsymbol{x} - \widetilde{\boldsymbol{x}}_t}{1-t} \right\|^2 \right]. \tag{1}$$

Generation is done by approximately integrating the ODE through a numerical ODE solver and the trained model. For example, Euler's method with M steps gives

$$\boldsymbol{x}_0 = \boldsymbol{\epsilon} \sim p_{\mathrm{noise}}, \qquad \boldsymbol{x}_1 = \boldsymbol{x}_0 + \int_0^1 \boldsymbol{v}_t(\boldsymbol{x}_t)\,dt \approx \boldsymbol{x}_0 + \frac{1}{M} \sum_{i=0}^{M-1} \frac{F_\theta(\boldsymbol{x}_{i/M}, \frac{i}{M}) - \boldsymbol{x}_{i/M}}{1 - \frac{i}{M}}. \tag{2}$$

2.2 Contrastive Self-supervised Representation Learning

For self-supervised representation learning, we consider contrastive objectives, which learn a representation by pulling together strong augmented views of the same image and pushing apart views of different images (Chen et al., 2020a; He et al., 2020). Thus, for a given image $\boldsymbol{x}_i$, we take its two

views $A_a(\boldsymbol{x}_i)$ and $A_b(\boldsymbol{x}_i)$ as a positive pair, and define its negatives as the set of augmented views of all other images in the dataset, $\mathcal{S}^- = \{A(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}_i\},\ A \sim \mathcal{A}\}$.

Let f_θ be an encoder that maps an image to a d-dimensional representation vector, and h_ψ a projection head that maps the representation to a space where the contrastive loss is applied. We denote by $E(\boldsymbol{x}) = h_\psi(f_\theta(\boldsymbol{x}))$ the representation for contrastive learning.

Conventional approaches of contrastive learning either require a large batch size (Chen et al., 2020a) or maintain a large memory bank (He et al., 2020). To avoid these, we follow the recent advances in optimizing a global contrastive objective (Yuan et al., 2022; Wei et al., 2024):

$$\mathcal{L}_{\mathrm{GCL}} \ = \ \mathbb{E}_{\boldsymbol{x}_i \sim \mathcal{D},\ A_a, A_b \sim \mathcal{A}} \Big[\ \ell\big(E(A_a(\boldsymbol{x}_i)),\ E(A_b(\boldsymbol{x}_i))\big)\ \Big], \tag{3}$$

$$\ell(\boldsymbol{u}, \boldsymbol{v}) \ = \ \tau \log Z(\boldsymbol{u}, \boldsymbol{v}), \qquad Z(\boldsymbol{u}, \boldsymbol{v}) \ = \ \mathbb{E}_{\boldsymbol{x}' \sim \mathcal{S}^-} \big[\exp\big((\mathrm{sim}(\boldsymbol{u}, E(\boldsymbol{x}')) - \mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}))/\tau\big)\big] .$$

To optimize this compositional objective, we use the SogCLR algorithm (Yuan et al., 2022). The key idea is to maintain a moving-average estimator of $Z(\boldsymbol{x}_i, \cdot)$ for each anchor image $\boldsymbol{x}_i$ using only mini-batch data and use it to compute a stochastic gradient estimator. We refer readers to Yuan et al. (2022) for more details of the algorithm.

3 Self-conditioned Generation on Self-Supervised Representation

In this section, we present the proposed framework SCION, including the model structure and the training objective and strategies.

3.1 The Unified Model

Since we model images directly in pixel space, we adopt the "Just Image Transformer" (JiT) architecture for image generation (Li & He, 2026). To integrate generation with representation learning, we partition JiT into two components: an encoder f_θ and a decoder g_ϕ, parameterized by θ and ϕ, respectively. Both f_θ and g_ϕ consist of multiple Transformer blocks. To process an image, we split it into non-overlapping patches, linearly embed the patches into a sequence of T tokens, and prepend a learnable `[CLS]` token.

One encoder, two purposes. The encoder processes two types of inputs for two distinct purposes. For flow matching, it encodes a noise-diffused image $\widetilde{\boldsymbol{x}}_t = t\boldsymbol{x} + (1-t)\boldsymbol{\epsilon}$. For contrastive learning, it encodes a strongly augmented, non-diffused image $A(\boldsymbol{x})$. To accommodate both types of inputs, we condition the encoder f_θ on the timestep $t \in [0, 1]$, where $t = 1$ corresponds to a non-diffused image. To enable semantically self-conditioned generation, we additionally condition the encoder on an embedding vector $\boldsymbol{c}$, whose construction is described below. We therefore write the encoder as $f_\theta(\boldsymbol{z} \mid t, \boldsymbol{c})$, where $\boldsymbol{z}$ denotes either a clean image $\boldsymbol{x}$, its augmented image $A(\boldsymbol{x})$ or a noise-diffused image $\widetilde{\boldsymbol{x}}_t$.

The encoder output also serves two different purposes. For contrastive learning, it should provide a compact semantic representation that is invariant to variations in appearance and spatial layout. For generation, it must preserve fine-grained visual details and spatial structure. To accommodate these different requirements within a single encoder, we use two readouts from its output: the `[CLS]` token embedding $\boldsymbol{e} \in \mathbb{R}^d$ used for representation learning, and the patch-token embeddings $\boldsymbol{h} \in \mathbb{R}^{T \times d}$ used for generation. For convenience, we introduce two readout operators, $\pi_{\boldsymbol{e}}$ and $\pi_{\boldsymbol{h}}$, which extract the `[CLS]` token embedding and the patch-token embeddings from $f_\theta(\boldsymbol{z} \mid t, \boldsymbol{c})$, respectively.

Two conditioning modes for two purposes. For generative modeling, we would like the encoder to process the noise-diffused image $\widetilde{\boldsymbol{x}}_t$ conditioned on a semantic representation $\boldsymbol{e}(\boldsymbol{x})$ of the corresponding clean image, namely, $f_\theta(\widetilde{\boldsymbol{x}}_t \mid t, \boldsymbol{e}(\boldsymbol{x}))$. However, the representation $\boldsymbol{e}(\boldsymbol{x})$ must itself be produced by the same encoder, creating an apparent circular dependency: the representation used to condition the encoder cannot simultaneously depend on that conditioned forward pass.

To break this dependency, we introduce a learnable null embedding $\boldsymbol{e}_\varnothing$. The semantic representation of a clean image is first obtained through a null conditioned encoding step, $\boldsymbol{e}(\boldsymbol{x}) = \pi_{\boldsymbol{e}} \circ f_\theta(\boldsymbol{x} \mid 1, \boldsymbol{e}_\varnothing)$. Likewise, for a strongly augmented view $A(\boldsymbol{x})$, its representation for contrastive learning is computed as $\boldsymbol{e}(A(\boldsymbol{x})) = \pi_{\boldsymbol{e}} \circ f_\theta(A(\boldsymbol{x}) \mid 1, \boldsymbol{e}_\varnothing)$. The resulting clean-image representation $\boldsymbol{e}(\boldsymbol{x})$ is then used as the semantic conditioning signal when processing the corresponding noise-diffused image, yielding

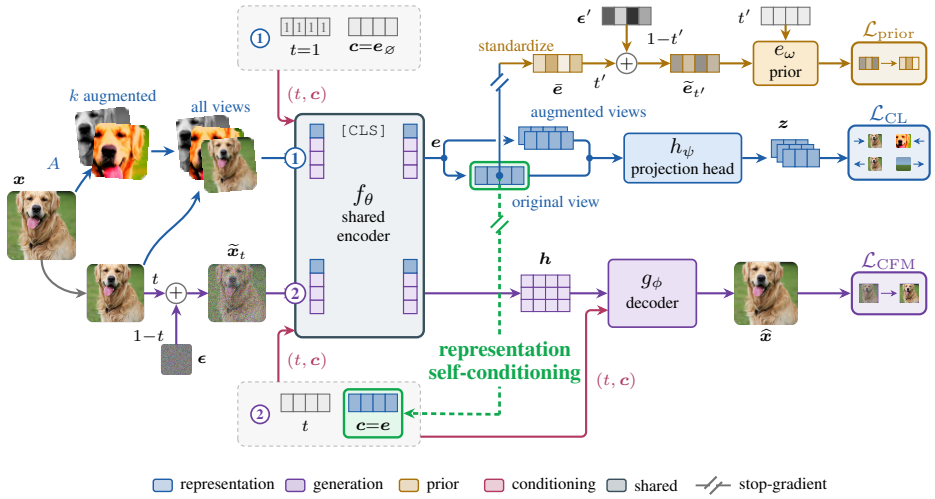

Figure 2: The network structure and information flow of **SCION.** One shared encoder f_θ is run twice per step. ① The *representation pass* (blue) encodes every view clean at $t{=}1$ under the learned null $\boldsymbol{e}_\varnothing$, and h_ψ maps the `[CLS]` readout to $\boldsymbol{z}$ for $\mathcal{L}_{\mathrm{CL}}$. ② The *generative pass* (violet) noises the original view and conditions on the detached $\mathbf{sg}(\boldsymbol{e})$ the first pass produced, the self-conditioning loop, drawn as the one dashed teal edge. Its patch tokens go to g_ϕ for $\mathcal{L}_{\mathrm{CFM}}$. The prior e_ω (amber) regresses the standardized embedding $\bar{e}$. A $/\!/$ across an edge marks a stop-gradient. The auxiliary $\mathcal{L}_{\mathrm{CBC}}$ term of Equation 4 is not drawn.

$\boldsymbol{h}_t = \pi_{\boldsymbol{h}} \circ f_\theta(\widetilde{\boldsymbol{x}}_t \mid t, \boldsymbol{e}(\boldsymbol{x}))$. Thus, the same encoder operates in two conditioning modes: null-conditioned encoding for representation learning and semantic-conditioned encoding for generation.

The decoder. Given a noise-diffused image $\widetilde{\boldsymbol{x}}_t$, the decoder takes the encoder's patch-token embeddings $\boldsymbol{h}_t = \pi_{\boldsymbol{h}} \circ f_\theta(\widetilde{\boldsymbol{x}}_t \mid t, \boldsymbol{e}(\boldsymbol{x}))$ as input. As with the encoder, we condition the decoder on both the timestep t and the clean-image representation $\boldsymbol{e}(\boldsymbol{x})$. The prediction is therefore given by $g_\phi(\boldsymbol{h}_t \mid t, \boldsymbol{e}(\boldsymbol{x}))$.

In summary, the unified model uses the same encoder in two complementary ways. For representation learning, an augmented image is encoded with null conditioning as $\boldsymbol{e}(A(\boldsymbol{x})) = \pi_{\boldsymbol{e}} \circ f_\theta(A(\boldsymbol{x}) \mid 1, \boldsymbol{e}_\varnothing)$, and its `[CLS]` token embedding is used for contrastive learning. For generation, the clean-image representation $\boldsymbol{e}(\boldsymbol{x}) = \pi_{\boldsymbol{e}} \circ f_\theta(\boldsymbol{x} \mid 1, \boldsymbol{e}_\varnothing)$ is used to condition both the encoder and decoder when processing $\widetilde{\boldsymbol{x}}_t$, with $\boldsymbol{h}_t = \pi_{\boldsymbol{h}} \circ f_\theta(\widetilde{\boldsymbol{x}}_t \mid t, \boldsymbol{e}(\boldsymbol{x}))$ and prediction $g_\phi(\boldsymbol{h}_t \mid t, \boldsymbol{e}(\boldsymbol{x}))$. In this way, a single encoder simultaneously supports semantic representation learning and semantically conditioned image generation. To implement conditioning in both the encoder and decoder, we adopt adaLN-Zero (Peebles & Xie, 2023). The proposed model structure of SCION is presented in Figure 2.

3.2 The Joint Training Objective

Below, we present each objective component and then combine them for joint training.

Contrastive loss. For a clean image $\boldsymbol{x}$, we generate two strong augmented copies $A_1(\boldsymbol{x})$ and $A_2(\boldsymbol{x})$. All 3 views are encoded by conditioning on $t = 1, \boldsymbol{c} = \boldsymbol{e}_\varnothing$. Following the practice of contrastive learning, the encoder's outputs are projected with a parameterized projection head h_ψ to produce the embeddings for contrastive learning, i.e., $\boldsymbol{z}^0 = h_\psi(f_\theta(\boldsymbol{x} \mid 1, \boldsymbol{e}_\varnothing))$, $\boldsymbol{z}^1 = h_\psi(f_\theta(A_1(\boldsymbol{x}) \mid 1, \boldsymbol{e}_\varnothing))$, and $\boldsymbol{z}^2 = h_\psi(f_\theta(A_2(\boldsymbol{x}) \mid 1, \boldsymbol{e}_\varnothing))$. Each of the three views will serve as a contrastive anchor for defining the contrastive loss:

$$\mathcal{L}_{\mathrm{GCL}}(\theta, \psi; \boldsymbol{x}) = \mathbb{E}_{A_1, A_2}\left[\frac{1}{6}\sum_{a,b=0, a\neq b}^{2} \ell\left(\boldsymbol{z}^a, \boldsymbol{z}^b\right)\right].$$

FM loss. For the clean image $\boldsymbol{x}$, we sample a noise-diffused image $\widetilde{\boldsymbol{x}}_t$ and define the CFM loss as:

$$\mathcal{L}_{\mathrm{CFM}}(\theta, \phi; \boldsymbol{x}) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\left\| \frac{g_\phi(\pi_{\boldsymbol{h}} \circ f_\theta(\widetilde{\boldsymbol{x}}_t \mid t, \boldsymbol{e}(\boldsymbol{x})) \mid t, \boldsymbol{e}(\boldsymbol{x})) - \widetilde{\boldsymbol{x}}_t}{1-t} - \frac{\boldsymbol{x} - \widetilde{\boldsymbol{x}}_t}{1-t} \right\|^2.$$

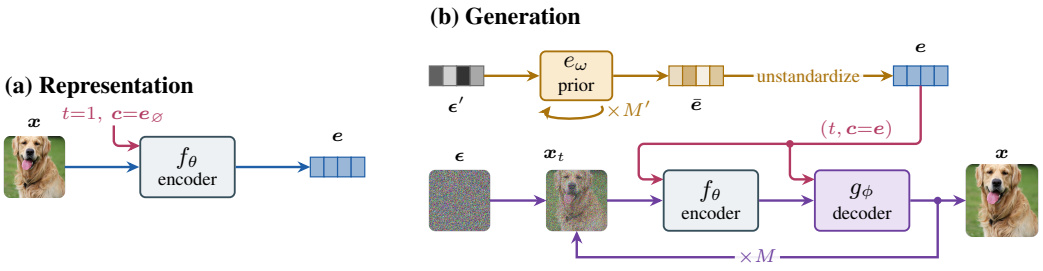

Figure 3: **How SCION is used. (a)** A clean image is encoded at $t{=}1$ with the learned null conditioning $\boldsymbol{e}_\varnothing$, reading out the representation $\boldsymbol{e}$. **(b)** To generate, the prior e_ω is integrated from noise to draw a standardized embedding $\bar{\boldsymbol{e}}$, which is unstandardized to an $\boldsymbol{e}$ that conditions the image ODE at every step. No label enters either path.

Learning the Representation Prior. During training, the generative model is conditioned on the representation $\boldsymbol{e}(\boldsymbol{x})$ of the input image. When generating new images at inference time, however, this representation is unavailable. We therefore learn a prior over representations using a flow matching model e_ω. For each image $\boldsymbol{x}$, we construct a standardized embedding $\bar{\boldsymbol{e}}(\boldsymbol{x}) = (\boldsymbol{e}(\boldsymbol{x}) - \boldsymbol{\mu})/\boldsymbol{\sigma}$ and its noisy counterpart $\tilde{\boldsymbol{e}}_{t'}(\boldsymbol{x}) = t'\bar{\boldsymbol{e}}(\boldsymbol{x}) + (1-t')\boldsymbol{\epsilon}'$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ denote the embedding mean and standard deviation vectors, division is elementwise, and $t' \in [0,1]$ and $\boldsymbol{\epsilon}'$ are a sampled timestep and noise vector, respectively. Given $\tilde{\boldsymbol{e}}_{t'}(\boldsymbol{x})$ and t', the model e_ω predicts the clean standardized embedding $\bar{\boldsymbol{e}}(\boldsymbol{x})$. Following CFM, we define the representation prior FM loss as

$$\mathcal{L}_{\text{prior}}(\omega; \boldsymbol{x}) = \mathbb{E}_{t',\boldsymbol{\epsilon}'} \left\| \frac{e_\omega(\tilde{\boldsymbol{e}}_{t'}(\boldsymbol{x}), t') - \tilde{\boldsymbol{e}}_{t'}(\boldsymbol{x})}{1-t'} - \frac{\bar{\boldsymbol{e}}(\boldsymbol{x}) - \tilde{\boldsymbol{e}}_{t'}(\boldsymbol{x})}{1-t'} \right\|^2 .$$

Conditioned-Branch Consistency Loss. The shared encoder operates in two conditioning modes: null conditioning for representation learning and representation conditioning for generation. Even for a clean image, these modes may produce different outputs, namely $f_\theta(\boldsymbol{x} \mid 1, \boldsymbol{e}_\varnothing)$ and $f_\theta(\boldsymbol{x} \mid 1, \boldsymbol{e}(\boldsymbol{x}))$. Moreover, when processing a noisy image $\widetilde{\boldsymbol{x}}_t$ for generation, the encoder should produce representations that remain semantically consistent with the clean-image representation $\boldsymbol{e}(\boldsymbol{x}) = \pi_{\boldsymbol{e}} \circ f_\theta(\boldsymbol{x} \mid 1, \boldsymbol{e}_\varnothing)$. Such consistency keeps the `[CLS]` token informative and, as the patch tokens attend to it in the encoder, shapes the patch features fed to the decoder. Since the preceding losses do not explicitly enforce this consistency, we introduce the conditioned-branch consistency (CBC) loss:

$$\mathcal{L}_{\text{CBC}}(\theta; \boldsymbol{x}) = -\mathbb{E}_{t,\boldsymbol{\epsilon}} \operatorname{sim}\big(\pi_{\boldsymbol{e}} \circ f_\theta(\widetilde{\boldsymbol{x}}_t \mid t, \boldsymbol{e}(\boldsymbol{x})), \boldsymbol{e}(\boldsymbol{x})\big).$$

Combining these losses yields the final joint objective:

$$\min_{\theta,\phi,\psi,\omega} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \Big[\lambda \mathcal{L}_{\text{GCL}}(\theta, \psi; \boldsymbol{x}) + \mathcal{L}_{\text{CFM}}(\theta, \phi; \boldsymbol{x}) + \mathcal{L}_{\text{prior}}(\omega; \boldsymbol{x}) + \eta \mathcal{L}_{\text{CBC}}(\theta; \boldsymbol{x}) \Big], \tag{4}$$

where η is a fixed hyper-parameter and λ is to balance between the global contrastive objective and the CFM objective.

3.3 Training Strategies and Sampling

To facilitate the training of the unified model by optimizing the joint objective, we employ the following two strategies.

Stop Gradients. For CFM, the gradient equivalence between conditional and marginal velocity regression established by Lipman et al. (2023) relies on target probability paths and velocity fields that are independent of the learnable parameters. In our setting, the conditioning representation $\boldsymbol{e}(\boldsymbol{x})$ depends on θ, so differentiating through it introduces an additional gradient path that is not covered by the standard equivalence argument (Appendix D). We therefore apply a stop-gradient operation to $\boldsymbol{e}(\boldsymbol{x})$ in the CFM loss, treating it as fixed conditioning during each update. We likewise detach $\boldsymbol{e}(\boldsymbol{x})$ in $\mathcal{L}_{\text{prior}}(\omega; \boldsymbol{x})$ and $\mathcal{L}_{\text{CBC}}(\theta; \boldsymbol{x})$. Consequently, only the contrastive loss backpropagates through the computation of $\boldsymbol{e}(\boldsymbol{x})$, while the CFM and CBC losses update the shared encoder through the f_θ. Algorithm 1 gives the complete joint training step.

Gradient Balancing. The losses $\mathcal{L}_{\text{CFM}}$ and $\mathcal{L}_{\text{GCL}}$ operate on different scales: the former is a weighted squared error in pixel space, whereas the latter is a contrastive objective governed by

the temperature τ. Consequently, the trade-off coefficient λ has no natural a priori scale. In our measurements, their gradient norms at the shared encoder differ by more than an order of magnitude, with $\|\nabla\mathcal{L}_{\mathrm{CL}}\| \approx 40\,\|\nabla\mathcal{L}_{\mathrm{CFM}}\|$. This ratio also varies with augmentation strength, learning rate, and batch size, making a fixed λ difficult to transfer across settings. We therefore adapt λ to balance the weighted gradient norms at a subset of shared encoder parameters $\varsigma \subseteq \theta$:

$$\lambda = \mathrm{clamp}\left(\frac{\mathrm{EMA}\big(\|\nabla_{\varsigma}\mathcal{L}_{\mathrm{CFM}}\|\big) + \xi}{\mathrm{EMA}\big(\|\nabla_{\varsigma}\mathcal{L}_{\mathrm{GCL}}\|\big) + \xi}, \lambda_{\min},\, \lambda_{\max}\right). \tag{5}$$

Here, EMA denotes an exponential moving average, $\xi = 10^{-8}$ provides numerical stability, and $[\lambda_{\min}, \lambda_{\max}] = [10^{-3}, 1]$. We recompute the gradient norms and update their EMAs every 10 training steps. We choose ς as the parameters of the encoder's last MLP layer, the deepest layer whose parameters receive gradients from both losses. The clamped ratio tracks changes in the gradient imbalance.

In Appendix B, we present Algorithm 2 for the complete balancing procedure, and Algorithm 1 for the joint training.

Sampling at Inference. At inference time, we first generate a standardized embedding by integrating the representation prior's ODE, defined by the model e_{ω}, starting from embedding-space noise $\boldsymbol{\epsilon}'$. We then reverse the standardization to obtain the representation $\boldsymbol{e}$ and use it to condition the image-generation ODE, defined by the model $F_{\theta,\phi} = g_{\phi} \circ \pi_{\boldsymbol{h}} \circ f_{\theta}$ (see Figure 3(b)).

Representation Guidance. To enable classifier-free guidance (Ho & Salimans, 2021) with representation conditioning at inference, we replace the conditioning representation with a second learned null embedding $\boldsymbol{c}_{\varnothing}$ with probability 0.1 during training. This allows the same network to learn both representation-conditioned and unconditional generation. At inference time, we integrate the image-generation ODE using the guided velocity field. Let the predicted velocity be $\boldsymbol{v}_{\theta,\phi}(\boldsymbol{x}_t, t \mid \boldsymbol{c}) = (F_{\theta,\phi}(\boldsymbol{x}_t, t \mid \boldsymbol{c}) - \boldsymbol{x}_t)/(1-t)$:

$$\boldsymbol{v}_{\mathrm{cfg}} = \boldsymbol{v}_{\theta,\phi}(\boldsymbol{x}_t, t \mid \boldsymbol{c}_{\varnothing}) + s\left(\boldsymbol{v}_{\theta,\phi}(\boldsymbol{x}_t, t \mid \boldsymbol{e}) - \boldsymbol{v}_{\theta,\phi}(\boldsymbol{x}_t, t \mid \boldsymbol{c}_{\varnothing})\right), \qquad s \geq 1, \tag{6}$$

where s is the guidance scale. It is notable that the two null embeddings serve distinct purposes: $\boldsymbol{c}_{\varnothing}$ enables unconditional generation, whereas $\boldsymbol{e}_{\varnothing}$ allows the encoder to extract $\boldsymbol{e}(\boldsymbol{x})$ without conditioning on the representation itself.

4 Experiments

This section evaluates SCION in three settings. We first conduct a controlled comparison with baselines that share the same backbone, training budget, and flow-matching recipe, isolating the contribution of the proposed framework. We then compare with published unconditional pixel-space systems using their reported scores, providing a system-level reference. Finally, we ablate SCION to identify the contribution of each component and compare its representation with that of a contrastive-only model.

Baselines. We compare SCION with vanilla flow matching (FM) and four methods built on it: Dispersive Loss (Wang & He, 2025), REPA (Yu et al., 2025), iREPA (Singh et al., 2026), and RCG (Li et al., 2024a). Thus, the baselines span FM alone, self-contained feature regularization (Dispersive), external-feature alignment (REPA/iREPA), and external semantic conditioning (RCG).

Setup. All controlled methods use the same JiT-B/16 backbone (Li & He, 2026) with attention residuals (Appendix F), flow matching recipe, ImageNet-1K (256×256) training data (Deng et al., 2009; Russakovsky et al., 2015), and a total training budget of 400 epochs. The baselines that import semantics use a frozen DINOv2 ViT-B encoder (Oquab et al., 2024), pretrained on external data. REPA and iREPA retain their original split of 4 encoder and 8 decoder blocks, whereas Dispersive Loss and SCION use 8 encoder and 4 decoder blocks. We split RCG's training budget between its generator and prior, training for 200 epochs each. We also evaluate RCG with both its original prior architecture and ours: the original architecture performs better, so we report that result.

Evaluation. We evaluate generation with FID and Inception Score (IS) under the ADM protocol (Dhariwal & Nichol, 2021) on 50K samples. We evaluate representations using ImageNet top-1

Table 2: **Controlled comparison on ImageNet** 256×256**.** All rows use the JiT-B/16 recipe for 400 epochs.

		generation		*representation*	
Method	pretrained enc.	FID↓	IS↑	Acc.↑	transfer-8↑
class-conditional					
JiT	✗	24.93	63.6	5.30	30.10
JiT + Dispersive	✗	26.73	58.7	7.79	20.72
JiT + REPA	✓	27.92	58.0	60.80	69.01
JiT + iREPA	✓	27.01	62.1	61.86	74.14
class-unconditional					
JiT	✗	55.40	19.2	11.27	26.84
JiT + Dispersive	✗	54.14	19.7	7.48	22.33
JiT + REPA	✓	49.73	24.6	63.57	72.62
JiT + iREPA	✓	46.44	26.2	65.11	75.43
RCG (JiT)	✓	14.27	93.4	6.54	30.28
SCION	✗	**8.92**	**118.4**	**72.35**	**79.89**

Table 3: **Comparison with published unconditional pixel-space systems on ImageNet** 256×256**.** † retrieves real images; RG denotes representation guidance; SCION (B/16) uses 800 epochs on JiT-B/16 and SCION (L/16) uses 400 epochs on JiT-L/16.

Method	pretrained enc.	params	FID↓	IS↑
BigGAN	✗	∼70M	38.61	24.7
ADM	✗	554M	26.21	39.7
RCDM†	✓ DINO	—	19.0	51.9
IC-GAN†	✓ SwAV	∼75M	15.6	59.0
RCG (ADM)	✓ MoCo v3	∼617M	6.24	136.9
SCION (B/16)	✗	231M	7.67	127.6
SCION (B/16, RG)	✗	231M	3.51	178.7
SCION (L/16)	✗	559M	5.89	151.6
SCION (L/16, RG)	✗	559M	**3.47**	**201.7**

accuracy (*Acc.*) and the average accuracy across eight datasets for transfer learning (*transfer-8*; datasets listed in Appendix E.2). For both probes, we train a logistic-regression classifier on the frozen representations produced by the model. For the baselines, the representation is the mean pooling of patch token embeddings at the encoder–decoder split, or after the last block for vanilla JiT and RCG; for SCION, we use its `[CLS]` token at its own split. Training and evaluation details are in Appendix E, and architecture details in Appendix F.

4.1 Controlled matched-recipe comparison

Table 2 compares all methods under the matched recipe. SCION achieves the best performance on both generation and representation capabilities. On unconditional generation, it achieves 8.92 FID, greatly improving on the best self-contained baseline (54.14). It also improves on the strongest alignment baseline, iREPA (46.44), and on RCG (14.27), both of which import semantics from a pretrained DINOv2 encoder. Moreover, SCION surpasses the class-conditional baselines, whose best unguided FID is 24.93, without using labels or an external encoder. Notably, after only 200 epochs, SCION already leads every matched baseline and surpasses RCG's 400-epoch-equivalent checkpoint on all four metrics, despite RCG's additional frozen prior-training stage (Table 9, Appendix C.2).

External DINOv2 semantics improve the other baselines in different ways: iREPA transfers part of them into the generator's own features, reaching 65.11% Acc. at 46.44 FID, whereas RCG improves generation to 14.27 FID but reaches only 6.54% Acc. and 30.28% transfer-8. SCION achieves 72.35% Acc. and 79.89% transfer-8, exceeding every baseline. Together with Table 1, these results show that SCION is the only controlled method satisfying all four "*S*" requirements while also achieving the best generation and representation results.

Representation guidance. SCION's learned representation can also replace labels as a guidance signal (Equation 6). Table 4 compares SCION's generation performance under representation guidance. SCION reaches 4.20 FID, improving on RCG's 5.01 while conditioning on its own learned representation and sampling it from a jointly learned prior. The class-guided JiT and REPA rows are provided for reference: they use ground-truth labels at sampling time, which SCION never observes.

4.2 System-level comparison

Table 3 compares SCION at two different model scales with published unconditional pixel-space systems: BigGAN (Donahue & Simonyan, 2019), ADM (Dhariwal & Nichol, 2021), RCDM (Bordes et al., 2022), IC-GAN (Casanova et al., 2021), and RCG (Li et al., 2024a). SCION (B/16) uses JiT-B/16 as the backbone, while SCION (L/16) uses JiT-L/16. Unlike the controlled comparison, these reported results span different architectures, training budgets, and protocols, and should therefore be read as a system-level reference. At 231M parameters, with self-representation guidance SCION (B/16) reaches 3.51 FID and 178.7 IS after 800 epochs, which are better than RCG (ADM), despite using less than half of RCG's ∼617M inference-time parameters and requiring no pretrained encoder.

Table 4: **Guided generation without label supervision.** Each row is evaluated at its independently swept guidance scale. Rows in gray text use ground-truth class labels and are included for reference.

Method	pretrained enc.	FID↓	IS↑
guided on a representation			
SCION (ours)	✗	**4.20**	162.1
RCG (JiT)	✓	5.01	**164.1**
guided on a class label, for reference			
JiT	✗	3.94	266.7
JiT + REPA	✓	3.57	286.9

Table 5: **Building SCION from unconditional flow matching.** Cumulative JiT-B/16 construction at a matched 200-epoch budget. Each row adds the stated component. CL-only is a separately trained contrastive-only control and has no generative objective.

Components	FID↓	Acc.↑	transfer-8↑
CL only	—	71.76	77.68
FM	61.17	10.65	26.19
+ CL	57.40	65.43	76.69
+ gradient-norm balancing	54.79	65.49	75.95
+ self-conditioning + prior	11.66	70.97	78.86
+ CBC (**SCION**)	**10.10**	**71.32**	**79.73**

For SCION (L/16) with 559M parameters, which remains slightly smaller than RCG (ADM), we only train for 400 epochs. Under self-representation guidance it reaches 3.47 FID and 201.7 IS, significantly outperforming every previously published pixel-space system in the table. Moreover, representation guidance improves SCION over unguided variants. Table 10 in Appendix C provides the broader comparison including latent-space systems.

4.3 Ablations

To isolate the contribution of each component, we use the same JiT-B/16 recipe as the controlled comparison and train every model for 200 epochs.

From flow matching to SCION. We decompose SCION by sequentially adding its components to unconditional flow matching. Table 5 shows that contrastive learning first makes the representation useful, increasing Acc. from 10.65% to 65.43% and transfer-8 from 26.19% to 76.69%, while slightly improving FID from 61.17 to 57.40. Gradient-norm balancing further reduces FID to 54.79 while maintaining representation quality and avoiding manual selection of the contrastive weight. Adding representation self-conditioning with the jointly learned prior converts these representation gains into substantially better generation, reducing FID to 11.66. Finally, CBC completes SCION and reaches 10.10 FID, with 71.32% Acc. and 79.73% transfer-8. Appendix C.1 provides a more detailed ladder.

Comparison with contrastive-only training. Table 5 also compares with a contrastive-only control using the same 8-layer encoder, evaluated at 200 epochs. Its learning rate, temperature, and weight decay are tuned independently for contrastive learning. SCION reaches 71.32% Acc., close to the control's 71.76%, while attaining higher transfer-8 accuracy (79.73% versus 77.68%). Thus, joint training retains representation quality comparable to dedicated contrastive learning while also learning to generate. Appendix C.1 repeats this comparison at 200, 400, and 600 epochs.

Additional ablations: Qualitative results, and generator vs. prior. Appendix G shows generations from real-image representations and interpolations between them. Appendix C.4 shows that conditioning on the prior instead of the encoder's representations raises precision and lowers recall.

5 Conclusion

We presented SCION, a framework that jointly learns image generation and semantic representations in a single pixel-space model, trained from scratch without labels or pretrained components. By integrating contrastive learning with flow matching, representation self-conditioning, and a jointly learned embedding prior, SCION supports both representation extraction and unconditional image generation. Experiments on ImageNet demonstrate substantial improvements over matched generative baselines while retaining representation quality competitive with dedicated contrastive learning. These results show that strong generation and useful representations can be learned together in a self-contained, self-supervised training process. Future work will explore downstream applications and extensions to domains where labels and pretrained models are scarce.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of derivations and assist in implementing methods. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, design or provide feedback on research methodology or experiments, clean and reformat datasets, propose or refine hypotheses, or interpret results. Using them to generate synthetic datasets, assist with translation, and support qualitative and thematic data analysis is not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, draft parts of this paper, edit this paper to improve readability, and identify relevant literature. We have reviewed all AI-assisted work. We used an LLM mostly to improve the readability of the writing; we therefore checked its generated text for correctness and ensured that all claims and content remained truthful. All derivations were verified by the authors. We also checked and tested the code for correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Like other image-generation methods, SCION could be used to produce misleading or harmful imagery. We consider the potential for misuse of the models studied here to be limited by their current image fidelity, 256×256 resolution, and class-unconditional sampling without text-based or explicit identity control. These limitations do not eliminate risk: generated images and learned representations may reflect biases or sensitive content in the training data, and we have not evaluated training-image memorization or demographic fairness. By removing the need for labels and pretrained components, the method also lowers the barrier to training on new domains and could enable more capable models. Such extensions would require assessment of risks appropriate to their capabilities and intended use.

Training uses multiple NVIDIA H200 GPUs (Appendix E), with an associated energy cost. Our single-stage approach avoids separate pretraining of encoders, autoencoders, or tokenizers, although we do not establish a reduction in total compute or energy consumption relative to other approaches.

The authors declare no conflicts of interest.

REPRODUCIBILITY STATEMENT

We will publicly release the training and evaluation code, together with SCION's checkpoints, to facilitate reproduction of our results. Section 3 describes the model, objectives, and sampling procedure, and Appendix B provides pseudocode for joint training and gradient-norm balancing. Architecture details are given in Appendix F. Appendix E documents hardware, preprocessing, augmentations, and optimization settings, with the main hyperparameters summarized in Table 12. Appendix E.2 specifies checkpoint selection, sampling and guidance settings, and the generation and representation evaluation protocols.

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=li7qeBbCR1t.

Fan Bao, Chongxuan Li, Jiacheng Sun, and Jun Zhu. Why are conditional generative models better than unconditional ones? In *NeurIPS 2022 Workshop on Score-Based Methods*, 2022a. URL https://openreview.net/forum?id=sbDyvrvvKn7.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=p-BhZSz59o4.

Andreas Blattmann, Robin Rombach, Kaan Oktay, Jonas Müller, and Björn Ommer. Retrieval-augmented diffusion models. *NeurIPS*, 35:15309–15324, 2022.

Florian Bordes, Randall Balestriero, and Pascal Vincent. High fidelity visualization of what your self-supervised representation knows about. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=urfWb7VjmL.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision Transformers. In *ICCV*, pp. 9650–9660, 2021.

Arantxa Casanova, Marlene Careil, Jakob Verbeek, Michal Drozdzal, and Adriana Romero Soriano. Instance-conditioned GAN. *NeurIPS*, 34:27517–27529, 2021.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image Transformer. In *CVPR*, 2022.

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

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pp. 1597–1607. PMLR, 2020a. URL http://proceedings.mlr.press/v119/chen20j.html.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big Self-Supervised Models are Strong Semi-Supervised Learners, October 2020b. URL http://arxiv.org/abs/2006.10029.

Xinlei Chen, Zhuang Liu, Saining Xie, and Kaiming He. Deconstructing denoising diffusion models for self-supervised learning. In *International Conference on Learning Representations*, volume 2025, pp. 55458–55472, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/8b52c29d1a77e3fcc6c8f90fa0ac0e8e-Paper-Conference.pdf.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, 2014. doi: 10.1109/CVPR.2014.461.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow Matching in Latent Space, July 2023. URL http://arxiv.org/abs/2307.08698.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *NeurIPS*, 2021.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *NeurIPS*, 32, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 2018.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming Transformers for high-resolution image synthesis. In *CVPR*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-First International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=FPnUhsQJ5B.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv:1803.07728*, 2018.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv:2006.07733*, 2020.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9726–9735, 2020. doi: 10.1109/CVPR42600.2020.00975. URL https://openaccess.thecvf.com/content_CVPR_2020/html/He_Momentum_Contrast_for_Unsupervised_Visual_Representation_Learning_CVPR_2020_paper.html.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for Transformers. In *Findings of EMNLP*, 2020.

Karl Heun. Neue Methoden zur approximativen Integration der Differentialgleichungen einer unabhängigen Veränderlichen. *Z. Math. Phys*, 45:23–38, 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. *NeurIPS*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshops*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple diffusion: End-to-end diffusion for high resolution images. *ICML*, 2023.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler Diffusion (SiD2): 1.5 FID on ImageNet512 with pixel-space diffusion. In *CVPR*, 2025.

Vincent Tao Hu, David W. Zhang, Yuki M. Asano, Gertjan J. Burghouts, and Cees G. M. Snoek. Self-guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18413–18422, June 2023.

Drew A Hudson, Daniel Zoran, Mateusz Malinowski, Andrew K Lampinen, Andrew Jaegle, James L McClelland, Loic Matthey, Felix Hill, and Alexander Lerchner. SODA: Bottleneck diffusion models for representation learning. In *CVPR*, 2024.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/ioffe15.html.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals, 2026. URL https://arxiv.org/abs/2603.15031.

Theodoros Kouzelis, Efstathios Karypidis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. Boosting generative image modeling via joint image-feature synthesis. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=i4qAfV04rZ.

Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification., 2017.

Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. In *Second Workshop on Fine-Grained Visual Categorization*, 2013.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *NeurIPS*, 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. In *NeurIPS*, 2024.

Samuel Lavoie, Michael Noukhovitch, and Aaron Courville. Compositional discrete latent code for high fidelity, productive diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1iSnpztjbD.

Fei-Fei Li, Marco Andreeto, Marc'Aurelio Ranzato, and Pietro Perona. Caltech 101, April 2022.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *CVPR*, 2026.

Tianhong Li, Huiwen Chang, Shlok Kumar Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. MAGE: Masked generative encoder to unify representation learning and image synthesis. In *CVPR*, 2023. URL https://arxiv.org/pdf/2211.09117.pdf.

Tianhong Li, Dina Katabi, and Kaiming He. Return of unconditional generation: A self-supervised representation generation method. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 125441–125468. Curran Associates, Inc., 2024a. doi: 10.52202/079017-3985. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e304d374c85e385eb217ed4a025b6b63-Paper-Conference.pdf.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024b.

Zinan Lin, Enshu Liu, Xuefei Ning, Junyi Zhu, Wenyu Wang, and Sergey Yekhanin. Latent zoning network: A unified principle for generative modeling, representation learning, and classification, 2025. URL https://arxiv.org/abs/2509.15591.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Deyuan Liu, Peng Sun, Xufeng Li, and Tao Lin. Efficient generative model training via embedded representation warmup, 2025. URL https://arxiv.org/abs/2504.10188.

Steven Liu, Tongzhou Wang, David Bau, Jun-Yan Zhu, and Antonio Torralba. Diverse image generation via self-conditioned GANs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14286–14295, 2020.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=XVjTT1nw5z.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Mario Lucic, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. In *International Conference on Machine Learning*, pp. 4183–4192. PMLR, 2019.

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*, 2024.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.

Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv:2103.03841*, 2021.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, December 2008.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pp. 69–84. Springer, 2016.

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. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023. doi: 10.1109/ICCV51070.2023.00387. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10619–10629, 2022.

Kai Qiu, Xiang Li, Jason Kuen, Hao Chen, Xiaohao Xu, Jiuxiang Gu, Yinyi Luo, Bhiksha Raj, Zhe Lin, and Marios Savvides. Robust latent matters: Boosting image generation with sampling error synthesis. *arXiv preprint arXiv: 2503.08354*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022. doi: 10.1109/CVPR52688.2022.01042. URL https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *NeurIPS*, 29, 2016.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: an open large-scale dataset for training next generation image-text models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Noam Shazeer. GLU variants improve Transformer. *arXiv:2002.05202*, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=y0UxFtXqXf.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Changyao Tian, Chenxin Tao, Jifeng Dai, Hao Li, Ziheng Li, Lewei Lu, Xiaogang Wang, Hongsheng Li, Gao Huang, and Xizhou Zhu. ADDP: Learning general representations for image recognition and generation with alternating denoising diffusion process. In *ICLR*, 2024. URL https://openreview.net/forum?id=cMPm8YFXZe.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Luting Wang, Yang Zhao, Zijian Zhang, Jiashi Feng, Si Liu, and Bingyi Kang. Image understanding makes for a good tokenizer for image generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 31015–31035. Curran Associates, Inc., 2024. doi: 10.52202/079017-0976. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/37611b0fc2b65cdbb60865af5f6cf453-Paper-Conference.pdf.

Runqian Wang and Kaiming He. Diffuse and disperse: Image generation with representation regularization, 2025. URL https://arxiv.org/abs/2506.09027.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. PixNerd: Pixel neural field diffusion. *arXiv:2507.23268*, 2025a.

Yingheng Wang, Yair Schiff, Aaron Gokaslan, Weishen Pan, Fei Wang, Christopher De Sa, and Volodymyr Kuleshov. InfoDiffusion: representation learning using information maximizing diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

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.

Xiyuan Wei, Fanjiang Ye, Ori Yonay, Xingyu Chen, Baixi Sun, Dingwen Tao, and Tianbao Yang. FastCLIP: A suite of optimization techniques to accelerate CLIP training with limited resources. *arXiv preprint arXiv: 2407.01445*, 2024.

Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, Jian Yang, Ming-Ming Cheng, and Xiang Li. Representation entanglement for generation: Training diffusion transformers is much easier than you think. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=koEALFNBj1.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *CVPR*, 2025.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, volume 2025, pp. 87400–87442, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d9e42b4d7163931f3689d6d6fbaa11d0-Paper-Conference.pdf.

Zhuoning Yuan, Yuexin Wu, Zi-Hao Qiu, Xianzhi Du, Lijun Zhang, Denny Zhou, and Tianbao Yang. Provable stochastic optimization for global contrastive learning: Small batch does not harm performance. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 25760–25782. PMLR, 2022. URL https://proceedings.mlr.press/v162/yuan22b.html.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, 2019.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Shaofeng Zhang, Xuanqi Chen, Ning Liao, Haoxiang Zhao, Xiaoxing Wang, Haoru Tan, Sitong Wu, Xiaosong Jia, Qi Fan, and Junchi Yan. Repulsor: Accelerating generative modeling with a contrastive memory bank, 2025. URL https://arxiv.org/abs/2512.08648.

Zijian Zhang, Zhou Zhao, and Zhijie Lin. Unsupervised representation learning from pre-trained diffusion probabilistic models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders, 2025. URL https://arxiv.org/abs/2510.11690.

Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT pre-training with online tokenizer. *arXiv:2111.07832*, 2021.

A Extended related work

Semantic conditioning for generation. Conditional GANs established that class information can substantially improve image synthesis (Mirza & Osindero, 2014; Brock et al., 2019). Thus, although DDPM initially demonstrated high-quality unconditional synthesis (Ho et al., 2020), it was soon followed by Improved DDPM incorporating class labels into diffusion (Nichol & Dhariwal, 2021). Classifier guidance further strengthened the effect of this semantic signal (Dhariwal & Nichol, 2021), while classifier-free guidance subsequently removed the need for a separate classifier, retaining label conditioning (Ho & Salimans, 2021). Together, these developments made semantics a central lever for generation, but supplied them directly through annotations.

Alongside this use of labels, efforts to reduce computation brought another source of semantics into the pipeline. Latent diffusion and flow models generate through pretrained autoencoders (Rombach et al., 2022; Dao et al., 2023), a setting later continued by transformer generators such as DiT and SiT (Peebles & Xie, 2023; Ma et al., 2024). Although introduced for compression, these autoencoders can inherit semantic information from their training objectives. VQGAN (Esser et al., 2021), for example, uses a perceptual loss (Zhang et al., 2018) based on supervised features (Simonyan & Zisserman, 2014). The semantic information carried by these tokenizers can therefore shape the representations learned on top of them (Chen et al., 2025b) and also benefit generation beyond reconstruction fidelity (Wang et al., 2024).

Importing representations for generation. Beyond the semantics inherited from autoencoders, generators can also draw on representations supplied explicitly by pretrained encoders. Some align autoencoder latents with frozen features (Yao et al., 2025; Qiu et al., 2025), while representation autoencoders generate directly in a pretrained encoder's latent space (Zheng et al., 2025). Others bring the same semantic signal into the generator itself through feature alignment, as in REPA and iREPA (Yu et al., 2025; Singh et al., 2026). Since fixed alignment can conflict with generative detail, subsequent work refines how pretrained features interact with generative training (Wang et al., 2025b; Wu et al., 2025; Liu et al., 2025). ReDi instead models images and pretrained features jointly (Kouzelis et al., 2025).

Beyond defining the latent space or guiding internal features, representations can serve directly as conditions. RCG and DLC take this route, conditioning on pretrained embeddings or discrete codes derived from them, and training separate models to sample them (Li et al., 2024a; Lavoie et al., 2025). Related retrieval and instance-conditioned methods obtain conditions from real images (Blattmann et al., 2022; Bordes et al., 2022; Casanova et al., 2021), while earlier approaches replace labels with clusters or proxy labels (Lucic et al., 2019; Liu et al., 2020; Bao et al., 2022a; Hu et al., 2023). These approaches show that semantic conditioning need not rely on class labels, but importing representations introduces dependencies on pretrained components or separate training stages. Learning these representations within the generator itself would therefore reduce the pipeline's external dependencies.

Self-contained generation. The pretrained components used by many of these approaches motivate a complementary direction that removes such dependencies. Pixel-space diffusion and flow models avoid the upstream autoencoder (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023), and recent recipes have made this setting increasingly competitive at high resolution (Hoogeboom et al., 2023; 2025; Chen et al., 2025a; Wang et al., 2025a; Li & He, 2026). Operating on pixels therefore removes one dependency, although it does not itself replace the semantic signal supplied by class labels. Dispersive Loss and Repulsor extend this effort to representation regularization, pushing

Table 6: **Comparison under the four S requirements**, extending Table 1 to the comparable systems tabulated in this appendix. **Self-contained** means no pretrained component, including a latent autoencoder or tokenizer; **self-supervised** means no label supervision anywhere in the pipeline, including through pretrained components; **single-model** means that the representation is produced by the generator's own network; and **single-stage** means that all components are learned jointly from scratch. Capabilities report a learned and evaluated **Repr**esentation and unconditional **Synth**esis with no source image to encode. All pipelines are as published, except REPA and Dispersive, which we combine with the JiT recipe removing the latent model dependency.

	four S requirements				*capabilities*	
Method	Self-cont.	Self-sup.	Single-mod.	Single-stage	Repr.	Synth.
Generative recipes						
DiT / SiT (Peebles & Xie, 2023; Ma et al., 2024)	✗	✗	–	✗	✗	✓
JiT (Li & He, 2026)	✓	✗	–	✓	✗	✓
JiT + REPA (Yu et al., 2025)	✗	✗	✗	✗	✗	✓
JiT + Dispersive (Wang & He, 2025)	✓	✗	✓	✓	✗	✓
ReDi (Kouzelis et al., 2025)	✗	✗	✗	✗	✗	✓
Semantic conditions						
RCG (ADM) (Li et al., 2024a)	✗	✓	✗	✗	✗	✓
DLC (Lavoie et al., 2025)	✗	✗	✗	✗	✗	✓
Joint representation and generation						
MAGE (Li et al., 2023)	✗	✗	✓	✗	✓	✓
ADDP (Tian et al., 2024)	✗	✗	✓	✗	✓	✓
PDAE (Zhang et al., 2022)	✗	✓	✗	✗	✓	✓
DiffAE (Preechakul et al., 2022)	✓	✓	✗	✗	✓	✓
SODA (Hudson et al., 2024)	✓	✓	✗	✓	✓	✗
SCION (ours)	✓	✓	✓	✓	✓	✓

apart intermediate features without a pretrained representation encoder (Wang & He, 2025; Zhang et al., 2025). In particular, Dispersive Loss finds positive-pair regularization sensitive to noisy-view construction and therefore retains only contrastive repulsion, arguing that denoising supplies attraction. The features nevertheless remain auxiliary regularizers, without downstream representation evaluation or explicit semantic conditioning, and the high-quality ImageNet settings retain class labels.

Joint generation and representation learning. Self-supervised learning offers a source of such semantics through pretext tasks (Noroozi & Favaro, 2016; Gidaris et al., 2018), masked modeling (Bao et al., 2022b; He et al., 2022; Zhou et al., 2021), self-distillation (Grill et al., 2020; Caron et al., 2021), and contrastive objectives (van den Oord et al., 2018; Chen et al., 2020a; He et al., 2020). The latter learn invariances by matching augmented views of the same image while separating different images. Although SimCLR benefits from large batches (Chen et al., 2020b), SogCLR reduces this requirement through a moving-average approximation to a global contrastive objective (Yuan et al., 2022). These objectives provide useful representations, but on their own do not define an image generator. Joint methods therefore seek to combine the two capabilities. MAGE learns and evaluates both in one transformer (Li et al., 2023), but operates on pretrained VQGAN tokens (Esser et al., 2021), thus retaining the inherited perceptual supervision discussed above. ADDP and LZN follow a similar joint learning goal through tokenized or modular latent pipelines (Tian et al., 2024; Lin et al., 2025).

A different route makes the learned representation a condition for a diffusion decoder. DiffAE learns this code through reconstruction, then fits a latent prior for unconditional sampling (Preechakul et al., 2022). InfoDiffusion likewise conditions diffusion on learned codes, sampled from a prior or a separately trained prior model (Wang et al., 2023). SODA moves from reconstruction to cross-view generation, using a source view's code to generate a related view (Hudson et al., 2024). All three methods produce their representations with an encoder separate from the diffusion decoder, and SODA additionally requires a source image for synthesis. SCION follows the joint learning goal while bringing the representation into the pixel-space generator itself, where it is learned, evaluated, and used to condition synthesis. A prior trained in the same run supplies conditions without a source image, combining the four requirements summarized in Table 6 and defined in Section 1.

B ALGORITHMS

We give pseudocode for the two procedures of Section 3.

Algorithm 1 is a single training step. From every training image it forms $k_O = 1$ original and $k_A = 2$ augmented images, runs one encoder-only pass over all of them at $t = 1$ under the null conditioning $\boldsymbol{e}_\varnothing$ to obtain the embeddings $\boldsymbol{e}^{(k)}$. It then reuses $\mathbf{sg}(\boldsymbol{e}^{(1)})$ both as the conditioning of the generative path and as the prior's target, adds the conditioned-branch consistency term on the noised input, and combines the four losses with the balanced weight λ before one backward pass and one optimizer step.

Algorithm 2 is the balancer that sets λ. Every U steps it reads the gradient norms of $\mathcal{L}_{\mathrm{CFM}}$ and $\mathcal{L}_{\mathrm{CL}}$ at the shared subset ς, the feed-forward submodule of the last encoder block, aggregates them across GPUs, and updates exponential moving averages whose clamped ratio becomes λ for this step and the next $U - 1$ steps. The two losses reach ς through separate forward passes, so their gradients are already computed up to ς by the step's own backward. Thus, on an update step, the only additional cost to compute the ratio is backpropagating each loss independently through this single MLP layer.

Algorithm 1 One training step of SCION with gradient-norm balancing.

Require: Shared encoder f_θ, generative decoder g_ϕ, contrastive head h_ψ, prior e_ω; original-image and augmented-image families $\mathcal{A}_{\mathrm{o}}, \mathcal{A}_{\mathrm{a}}$ with k_O and k_A images, $K = k_O + k_A$

Require: Representation null $\boldsymbol{e}_\varnothing$, generation null $\boldsymbol{c}_\varnothing$ with drop probability p; CBC weight η; optimizer $\mathcal{O}$; subroutine GRADBALANCE (Algorithm 2); stop gradient operator $\mathbf{sg}(\cdot)$

Minibatch and views. Draw $\mathcal{B} = \{\boldsymbol{x}_i\}_{i=1}^{B} \sim \mathcal{D}$. For each i, draw $A_1 \sim \mathcal{A}_{\mathrm{o}}$ and $A_2, \ldots, A_K \sim \mathcal{A}_{\mathrm{a}}$, and set $\boldsymbol{x}_i^{(k)} = A_k(\boldsymbol{x}_i)$.

Representation pass (clean, $t = 1$, null conditioning; encoder only, all K views in one forward).

$\boldsymbol{e}_i^{(k)} \;=\; \pi_{\boldsymbol{e}}\big(f_\theta(\boldsymbol{x}_i^{(k)} \mid 1,\, \boldsymbol{e}_\varnothing)\big)$

$\boldsymbol{z}_i^{(k)} \;=\; h_\psi(\boldsymbol{e}_i^{(k)}) / \|h_\psi(\boldsymbol{e}_i^{(k)})\|$

Write $\boldsymbol{e}_i = \boldsymbol{e}_i^{(1)}$ for the original image's embedding.

Generative pass (original image only, full network).

$\boldsymbol{\epsilon}_i \sim p_{\mathrm{noise}}, \quad \mathrm{logit}(t_i) \sim \mathcal{N}(\mu, \sigma^2), \quad \widetilde{\boldsymbol{x}}_{i,t} = t_i\, \boldsymbol{x}_i^{(1)} + (1 - t_i)\, \boldsymbol{\epsilon}_i$

$\boldsymbol{c}_i \leftarrow \boldsymbol{c}_\varnothing$ with probability p, else $\boldsymbol{c}_i \leftarrow \mathbf{sg}(\boldsymbol{e}_i)$

$\boldsymbol{e}_{\mathrm{gen}}(\widetilde{\boldsymbol{x}}_{i,t},\, t_i) = \pi_{\boldsymbol{e}}\big(f_\theta(\widetilde{\boldsymbol{x}}_{i,t} \mid t_i,\, \boldsymbol{c}_i)\big)$ and $\boldsymbol{h}_i = \pi_{\boldsymbol{h}}\big(f_\theta(\widetilde{\boldsymbol{x}}_{i,t} \mid t_i,\, \boldsymbol{c}_i)\big)$, one forward

$F_{\theta,\phi}(\widetilde{\boldsymbol{x}}_{i,t},\, t_i \mid \boldsymbol{c}_i) \;=\; g_\phi\big(\boldsymbol{h}_i \mid t_i,\, \boldsymbol{c}_i\big)$ ($\boldsymbol{x}$-prediction)

Prior pass (original image only, no image forward).

Update the EMA frame $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ from the original image; set $\bar{\boldsymbol{e}}_i = (\boldsymbol{e}_i - \boldsymbol{\mu})/\boldsymbol{\sigma}$.

$\boldsymbol{\epsilon}'_i \sim p_{\mathrm{noise}}, \quad \mathrm{logit}(t'_i) \sim \mathcal{N}(\mu', \sigma'^2), \quad \widetilde{\boldsymbol{e}}_{i,t'} = t'_i\, \bar{\boldsymbol{e}}_i + (1 - t'_i)\, \boldsymbol{\epsilon}'_i$

Evaluate the prior at $(\widetilde{\boldsymbol{e}}_{i,t'},\, t'_i)$.

Losses, as in Equation 4.

$\mathcal{L}_{\mathrm{CL}}$ from $\{\boldsymbol{z}_i^{(k)}\}$ with negatives gathered across GPUs, the multi-view form of Equation 3

$\mathcal{L}_{\mathrm{CFM}}$ on $F_{\theta,\phi}(\widetilde{\boldsymbol{x}}_{i,t},\, t_i \mid \boldsymbol{c}_i)$ against $\boldsymbol{x}_i^{(1)}$, Equation 1

$\mathcal{L}_{\mathrm{prior}}$ on $e_\omega(\widetilde{\boldsymbol{e}}_{i,t'},\, t'_i)$ against $\mathbf{sg}(\bar{\boldsymbol{e}}_i)$

$\mathcal{L}_{\mathrm{CBC}} = -\mathrm{sim}\big(\boldsymbol{e}_{\mathrm{gen}}(\widetilde{\boldsymbol{x}}_{i,t},\, t_i),\, \mathbf{sg}(\boldsymbol{e}_i)\big)$

Balance, backward and update.

$\lambda \leftarrow \mathrm{GRADBALANCE}(\mathcal{L}_{\mathrm{CFM}},\, \mathcal{L}_{\mathrm{CL}})$

One backward pass on $\mathcal{L}_{\mathrm{CFM}} + \lambda\, \mathcal{L}_{\mathrm{CL}} + \mathcal{L}_{\mathrm{prior}} + \eta\, \mathcal{L}_{\mathrm{CBC}}$, then $\mathcal{O}.\mathrm{step}()$

Algorithm 2 Gradient-norm balancer of Section 3.3. It keeps $\mathcal{L}_{\mathrm{CFM}}$ and $\mathcal{L}_{\mathrm{CL}}$ at comparable influence on the shared encoder by setting λ to the clamped ratio of their moving-average gradient norms at ς, which alleviates the need to tune λ per setting. It is called with the current losses and step number and returns the weight to use.

Require: Shared parameters subset $\varsigma \subseteq \theta$, the feed-forward of the last encoder block
Require: Scale λ_0, clamp $[\lambda_{\min}, \lambda_{\max}]$, update period U, decay range $[\beta_{\min}, \beta_{\max}]$, warmup W, small number ξ (e.g. 10^{-8}).
State: $\mathrm{EMA}_{\mathrm{CFM}} \leftarrow 0$, $\mathrm{EMA}_{\mathrm{CL}} \leftarrow 0$, $\lambda \leftarrow \lambda_0$, update count $k \leftarrow 0$.
Inputs at this call: the two losses, already computed, and the step index s.
if $s \bmod U \neq 0$ **then**
  return λ (cached; the step does no extra work)
end if
$k \leftarrow k + 1$
Backpropagate $\mathcal{L}_{\mathrm{CFM}}$ through the decoder and $\mathcal{L}_{\mathrm{CL}}$ through the head, each down to the output of ς. The two objectives reach ς through separate forward passes, so these are computed by the step's own backward pass.
Continue each adjoint through ς alone to obtain $\nabla_\varsigma \mathcal{L}_{\mathrm{CFM}}$ and $\nabla_\varsigma \mathcal{L}_{\mathrm{CL}}$.
Local squared norms $q_\bullet \leftarrow \|\nabla_\varsigma \mathcal{L}_\bullet\|^2$ for $\bullet \in \{\mathrm{CFM}, \mathrm{CL}\}$.
Reduce across GPUs, $Q_\bullet \leftarrow \textsc{all_reduce}(q_\bullet)$, and take global norms $n_\bullet \leftarrow \sqrt{Q_\bullet}$.
Decay warmup: $\beta \leftarrow \beta_{\min} + \min(1,\, s/W)\,(\beta_{\max} - \beta_{\min})$.
$\mathrm{EMA}_\bullet \leftarrow \beta\,\mathrm{EMA}_\bullet + (1-\beta)\,n_\bullet$.
$\lambda \leftarrow \lambda_0 \cdot \mathrm{clamp}\left(\dfrac{\mathrm{EMA}_{\mathrm{CFM}} + \xi}{\mathrm{EMA}_{\mathrm{CL}} + \xi},\ \lambda_{\min},\ \lambda_{\max}\right)$
return λ

C Additional Experimental Results

C.1 Detailed ablations

This section expands the construction ablation in Section 4.3. Table 7 decomposes our joint-training recipe. It first isolates adding the contrastive loss with different view splits to the flow matching model. It then cumulatively adds gradient-norm balancing, representation self-conditioning with the learned prior, and, lastly, the CBC loss. All results are reported using the JiT-B/16 recipe, trained for 200 epochs in the unconditional setting. Table 8 then tracks SCION and capacity-matched contrastive-only controls over training. All rows train the same 8-layer encoder from scratch, while the controls use learning rate, temperature, and weight decay tuned independently for contrastive learning. Each control is a single 600-epoch run with a cosine learning-rate schedule, evaluated at its 200-, 400-, and 600-epoch checkpoints.

Contrastive learning benefits both tasks before self-conditioning. Adding contrastive learning with one original and two augmented inputs substantially improves representation quality while modestly improving the still weak pre-prior generation baseline. FID decreases from 61.17 to 57.40, while Acc. rises from 10.65% to 65.43% and transfer-8 from 26.19% to 76.69%. Using only augmented inputs obtains similar representation quality but degrades FID to 70.46. From rung 2 on, the representation is read from a `[CLS]` token. With it, gradient-norm balancing reaches 54.79 FID, better than every fixed contrastive weight in rung 2 (56.00 to 60.25), without materially changing representation quality and without manual selection of the contrastive weight.

Self-conditioning is the generative lever. Adding representation self-conditioning together with the learned prior converts the improved representation into substantially better generation, reducing FID to 11.66 while improving Acc. and transfer-8 to 70.97% and 78.86%, respectively. Adding CBC completes SCION and further improves generation, reducing FID to 10.10.

Comparison with contrastive-only training. The control with one original and two augmented views achieves the highest Acc. at every budget, and the control with two augmented views also exceeds SCION at 400 and 600 epochs, whereas SCION achieves the highest transfer-8 score at every budget. Therefore, across the 200, 400, and 600-epoch budgets, SCION learns representations broadly comparable to those from dedicated contrastive training.

Table 7: **Detailed ablation ladder.** All rows use the JiT-B/16 architecture under the same recipe for 200 epochs and unconditional setting. Rungs 0 to 3 isolate the joint-training recipe before representation self-conditioning and the learned prior. Rung 4 adds both components, and rung 5 adds the conditioned-branch consistency loss $\mathcal{L}_{\mathrm{CBC}}$ to complete SCION. Rung 0 is our unconditional flow-matching reproduction. *views* gives the number of original and augmented inputs, respectively; *bal.* denotes gradient-norm balancing, which rescales the contrastive weight online.

			generation					*representation*	
Rung	views	λ_{CL}	FID↓	sFID↓	IS↑	Prec.↑	Rec.↑	Acc.↑	transfer-8↑
0. flow matching alone (JiT-B/16 unconditional)									
	$1+0$	—	61.17	9.05	17.53	.384	.559	10.65	26.19
1. add contrastive learning: which views?									
1i	$3+0$	0.025	59.08	9.25	17.96	.385	.571	33.74	50.93
1ii	$0+3$	0.025	70.46	16.89	15.18	.237	**.625**	65.30	77.36
1iii	$1+2$	0.025	57.40	9.65	18.72	.406	.599	65.43	76.69
2. sweep the contrastive weight									
2a	$1+2$	0.01	56.00	8.83	19.11	.413	.574	62.40	74.53
2b	$1+2$	0.025	57.18	8.83	19.27	.411	.579	65.47	76.00
2c	$1+2$	0.06	60.25	9.09	18.51	.389	.578	65.37	75.07
3. online gradient balancer instead of manual tuning									
	$1+2$	bal.	54.79	8.62	19.76	.425	.586	65.49	75.95
4. add representation self-conditioning and the prior									
	$1+2$	bal.	11.66	7.49	95.49	.680	.520	70.97	78.86
5. add CBC (final SCION)									
	$1+2$	bal.	**10.10**	**7.24**	**101.91**	**.715**	.497	**71.32**	**79.73**

Table 8: **Representation quality throughout training.** We compare SCION with capacity-matched contrastive-only controls trained from scratch. All rows use the same 8-layer encoder; the controls have no generative objective and are independently tuned for contrastive learning. During inference, the models use approximately the same number of parameters, 57.2M for SCION and 57.1M for the controls.

		Acc.↑			transfer-8↑		
Method	views	200	400	600	200	400	600
CL only	2 augmented	70.48	73.61	74.84	77.04	78.88	77.09
CL only	1 original + 2 augmented	**71.76**	**74.15**	**74.98**	77.68	78.66	76.61
SCION (ours)	1 original + 2 augmented	71.32	72.35	72.61	**79.73**	**79.89**	**79.87**

C.2 Controlled comparison at a shorter training budget

Table 9 repeats the matched comparison after 200 epochs. SCION already leads every single-run baseline on both generation and representation metrics, and exceeds RCG's 400-epoch-equivalent checkpoint on every reported metric.

C.3 Unconditional generation system-level comparison

Table 10 presents latent and pixel-space published unconditional ImageNet systems in a common dependency view. The lowest latent-space FIDs come from multi-stage pipelines built on pretrained autoencoders or tokenizers and, for ReDi and DLC, external DINOv2 features trained beyond ImageNet. Within the pixel-space block, SCION (L/16) reaches 5.89 FID without an autoencoder, tokenizer, external model, labels, or external data, and representation guidance reduces this to 3.47 using a prior learned in the same run. Since architectures, protocols, and training budgets differ across systems, the table should be read as a system-level comparison. Tables 2 and 4 give the controlled matched-recipe comparisons.

Table 9: **Controlled class-unconditional comparison at 200 epochs.** All methods use the same JiT-B/16 flow-matching recipe on ImageNet (256×256). The RCG row is a reference: its 200-epoch generator is preceded by a frozen 200-epoch prior stage, for 400 total epoch-equivalents.

Method	FID↓	IS↑	Acc.↑	transfer-8↑
JiT	61.17	17.5	10.65	26.19
JiT + Dispersive	58.68	18.0	8.88	24.06
JiT + REPA	53.07	23.0	63.94	73.59
JiT + iREPA	48.43	24.8	65.81	76.37
RCG (JiT; 400 total)	14.27	93.4	6.54	30.28
SCION (ours)	**10.10**	**101.9**	**71.32**	**79.73**

Table 10: **Unconditional ImageNet-1K (Deng et al., 2009)** 256×256 **generation systems and their dependencies.** FID and IS as reported by the corresponding work. *Lat.* names the pretrained VQGAN tokenizer (Esser et al., 2021) or latent autoencoder, SD-VAE[a] or an LDM autoencoder (LDM-AE) (Rombach et al., 2022), and *Ext.* any other pretrained model. *Lab.* marks label supervision anywhere in the pipeline, including the supervised perceptual loss (Zhang et al., 2018) inherited by VQGAN and SD-VAE. *External data* marks a dependency trained on data beyond ImageNet-1K. In particular, SD-VAE uses OpenImages (Krasin et al., 2017) and, per its model card, is fine-tuned on LAION-Aesthetics (Schuhmann et al., 2022) and an unreleased LAION-Humans subset, DINOv2 uses LVD-142M (Oquab et al., 2024), and CLIP uses WIT-400M (Radford et al., 2021). *Multi-stage* marks more than one separately trained component. *Params* count all inference-time models, including a latent autoencoder or tokenizer where applicable. † retrieves real images at sampling. * denotes DiT values reported by ReDi and ‡ values reported by RCG. (CFG) and (RG) denote classifier-free and representation guidance, respectively. Training budgets are not matched. SCION (B/16) is reported at 800 epochs and SCION (L/16) at 400 epochs.

	dependencies					*generation*		
Method	Lat.	Ext.	Lab.	External data	Multi-stage	Params	FID↓	IS↑
latent-space								
DiT-B/2* (Peebles & Xie, 2023)	SD-VAE	—	✓	✓	✓	179M	69.3	—
DiT-XL/2* (Peebles & Xie, 2023)	SD-VAE	—	✓	✓	✓	724M	44.6	—
ReDi (DiT-B/2) (Kouzelis et al., 2025)	SD-VAE	DINOv2	✓	✓	✓	179M	51.7	—
ReDi (DiT-B/2) (RG) (Kouzelis et al., 2025)	SD-VAE	DINOv2	✓	✓	✓	179M	47.3	—
ReDi (DiT-XL/2) (Kouzelis et al., 2025)	SD-VAE	DINOv2	✓	✓	✓	724M	25.1	—
ReDi (DiT-XL/2) (RG) (Kouzelis et al., 2025)	SD-VAE	DINOv2	✓	✓	✓	724M	22.6	—
MaskGIT‡ (Chang et al., 2022)	VQGAN	—	✓	✗	✓	227M	20.72	42.1
ADDP-B (Tian et al., 2024)	VQGAN	—	✓	✗	✓	176M	8.9	95.3
ADDP-L (Tian et al., 2024)	VQGAN	—	✓	✗	✓	439M	7.6	105.1
MAGE-B (Li et al., 2023)	VQGAN	—	✓	✗	✓	176M	8.67	94.8
MAGE-L (Li et al., 2023)	VQGAN	—	✓	✗	✓	439M	7.04	123.5
RDM-IN† (CFG) (Blattmann et al., 2022)	LDM-AE	CLIP	✓	✓	✓	400M	5.91	158.8
RCG (MAGE-B) (Li et al., 2024a)	VQGAN	MoCo v3	✓	✗	✓	239M	3.98	177.8
RCG (MAGE-L) (Li et al., 2024a)	VQGAN	MoCo v3	✓	✗	✓	502M	3.44	186.9
RCG (MAGE-B) (CFG) (Li et al., 2024a)	VQGAN	MoCo v3	✓	✗	✓	239M	3.19	212.6
RCG (MAGE-L) (CFG) (Li et al., 2024a)	VQGAN	MoCo v3	✓	✗	✓	502M	2.15	253.4
DLC512 (Lavoie et al., 2025)	SD-VAE	DINOv2	✓	✓	✓	1198M	2.00	260.7
DLC512 (CFG) (Lavoie et al., 2025)	SD-VAE	DINOv2	✓	✓	✓	1198M	1.59	255.4
pixel-space								
BigGAN‡ (Donahue & Simonyan, 2019)	—	—	✗	✗	✗	∼70M	38.61	24.7
ADM (Dhariwal & Nichol, 2021)	—	—	✗	✗	✗	554M	26.21	39.7
RCDM† (Bordes et al., 2022)	—	DINO	✗	✗	✓	—	19.0	51.9
IC-GAN† (Casanova et al., 2021)	—	SwAV	✗	✗	✓	∼75M	15.6	59.0
RCG (ADM) (Li et al., 2024a)	—	MoCo v3	✗	✗	✓	∼617M	6.24	136.9
SCION (B/16)	—	—	✗	✗	✗	231M	7.67	127.6
SCION (B/16) (RG)	—	—	✗	✗	✗	231M	3.51	178.7
SCION (L/16)	—	—	✗	✗	✗	559M	5.89	151.6
SCION (L/16) (RG)	—	—	✗	✗	✗	559M	3.47	201.7

[a] SD-VAE denotes the commonly used public `sd-vae-ft-ema` checkpoint.

Table 11: **Oracle conditioning against the learned prior.** SCION (B/16) at 400 epochs, unguided, 50K samples per arm. The *oracle* arm (†, semi-parametric) replaces the prior sample with the representation of a real training image, one sample per representation. Each arm is scored against 10K images from the conditioning set and from a disjoint set of training images. With the same generator, real representations give higher recall and prior samples higher precision.

conditioning	reference	Prec.↑	Rec.↑
prior e_ω	conditioning set	**.718**	.524
	disjoint set	**.722**	.510
oracle repr.†	conditioning set	.645	**.828**
	disjoint set	.626	**.664**

C.4 Analysis of the representation prior

At inference, the generator is conditioned on a representation sampled from the prior e_ω, so sample quality depends on both the generator and the prior. To separate the two, we replace the prior sample with the representation $\boldsymbol{e}(\boldsymbol{x})$ of a real training image, which we call *oracle* conditioning. It requires real images at generation time, so it serves as a diagnostic rather than as a way to sample.

Protocol. All results use SCION (B/16) at 400 epochs, without guidance. We draw a conditioning set of 50 images per class, 50K in total, from the ImageNet training set with a fixed seed, and generate one image from the representation of each. The prior arm consists of 50K images generated by the same model from prior samples, namely the samples behind the 8.92 FID in Table 2. We compute precision and recall (Appendix E.2) of both arms against two references: the conditioning set itself, and a disjoint set drawn in the same way from the remaining training images. For each reference we use a random 10K subset for precision and recall computation, matching what ADM does.

Real representations raise recall, prior samples raise precision. Table 11 reports the results. The prior arm serves as a control: its samples were generated from neither reference, and it scores nearly the same against both, with precision .718 and .722 and recall .524 and .510. The two references therefore behave alike for samples not generated from either, so a gap between them in the oracle arm reflects that its samples were generated from the conditioning images.

The oracle arm shows such a gap. Its recall is .828 against the conditioning set and .664 against the disjoint set, so the generated images cover the specific images they were conditioned on better than other images from the same distribution. The representation thus carries image-specific content into the generated image.

Against the disjoint set, where the oracle arm has no such advantage, it still reaches higher recall than the prior arm, .664 against .510, at lower precision, .626 against .722. The two arms share the generator and differ only in the conditioning vectors, so this difference is due to the prior: conditioned on real representations, the samples cover more of the reference images, while conditioned on prior samples, a larger fraction of them falls inside the reference manifold.

D Flow Matching equivalence under a learned conditioning

This appendix explains why the shared embedding is passed through stop-gradient before conditioning the generative and prior branches.

Our representation-conditioned generator uses the Conditional Flow Matching (CFM) objective (Lipman et al., 2023) for training. Since the conditioning embedding is itself produced by the shared encoder, we need to ensure that optimizing this conditional objective is still equivalent to optimizing the corresponding marginal Flow Matching objective. We will present the argument for the generation path, but the same reasoning applies to the representation prior.

At a given training step, we consider a sampled pair $(\boldsymbol{x}, \boldsymbol{c})$, where $\boldsymbol{x}$ is a clean data sample and $\boldsymbol{c} = \boldsymbol{c}_\theta(\boldsymbol{x})$ is the embedding produced by the shared encoder for that same sample. Following the standard FM setup, let

$$\boldsymbol{x} \sim p_{\text{data}}, \qquad \boldsymbol{\epsilon} \sim p_{\text{noise}}, \qquad \boldsymbol{c}_\theta(\boldsymbol{x}) = \pi_{\boldsymbol{e}} \circ f_\theta(\boldsymbol{x} \mid 1, \boldsymbol{e}_\varnothing), \tag{7}$$

and define the bridge state and conditional velocity by

$$\boldsymbol{x}_t = t\boldsymbol{x} + (1-t)\boldsymbol{\epsilon}, \qquad \boldsymbol{v}_t = \frac{d\boldsymbol{x}_t}{dt} = \boldsymbol{x} - \boldsymbol{\epsilon}, \tag{8}$$

where $\boldsymbol{x}$ is the clean data endpoint, $\boldsymbol{\epsilon}$ is the noise endpoint, and $\boldsymbol{x}_t$ is the intermediate point on the bridge between them. This is the conditional path written $\widetilde{\boldsymbol{x}}_t$ in Section 2.1; we drop the tilde in this appendix. For a generic intermediate location $\boldsymbol{u}$, the marginal conditional velocity is

$$\boldsymbol{v}_t^\star(\boldsymbol{u} \mid \boldsymbol{c}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon}}[\boldsymbol{v}_t \mid \boldsymbol{x}_t = \boldsymbol{u}, \boldsymbol{c}, t]\,. \tag{9}$$

This is the average bridge velocity among all sampled pairs $(\boldsymbol{x}, \boldsymbol{\epsilon})$ whose interpolant passes through the same intermediate point $\boldsymbol{u}$ under the same condition $\boldsymbol{c}$ at time t.

The network predicts the clean image, $F_\theta(\boldsymbol{x}_t, t \mid \boldsymbol{c}) = g_\phi \circ \pi_{\boldsymbol{h}} \circ f_\theta(\boldsymbol{x}_t, t \mid \boldsymbol{c})$, where we write θ for all generator parameters, including the decoder's ϕ, and the velocity model is the velocity it implies,

$$\boldsymbol{v}_\theta(\boldsymbol{x}_t, t \mid \boldsymbol{c}) = \frac{F_\theta(\boldsymbol{x}_t, t \mid \boldsymbol{c}) - \boldsymbol{x}_t}{1-t}. \tag{10}$$

The CFM and FM objectives for the generation path are

$$\mathcal{L}_{\mathrm{CFM}}(\theta; \boldsymbol{c}_\theta) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},t}\left[\|\boldsymbol{v}_\theta(\boldsymbol{x}_t, t \mid \boldsymbol{c}) - \boldsymbol{v}_t\|^2\right], \tag{11}$$

$$\mathcal{L}_{\mathrm{FM}}(\theta; \boldsymbol{c}_\theta) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},t}\left[\|\boldsymbol{v}_\theta(\boldsymbol{x}_t, t \mid \boldsymbol{c}) - \boldsymbol{v}_t^\star(\boldsymbol{x}_t \mid \boldsymbol{c})\|^2\right]. \tag{12}$$

The following result follows the same decomposition as Theorem 2 of Lipman et al. (2023), but accounts for the fact that our conditioning variable is learned.

Corollary 1 (FM and CFM with learned conditioning) *Let*

$$m_t = \boldsymbol{v}_t^\star(\boldsymbol{x}_t \mid \boldsymbol{c}) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon}}[\boldsymbol{v}_t \mid \boldsymbol{x}_t, \boldsymbol{c}, t].$$

If the conditioning embedding $\boldsymbol{c}_\theta$ is differentiated, then

$$\mathcal{L}_{\mathrm{CFM}}(\theta; \boldsymbol{c}_\theta) = \mathcal{L}_{\mathrm{FM}}(\theta; \boldsymbol{c}_\theta) + R(\theta), \qquad R(\theta) = \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},t}\left[\|\boldsymbol{v}_t - m_t\|^2\right]. \tag{13}$$

Consequently,

$$\nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta; \boldsymbol{c}_\theta) = \nabla_\theta \mathcal{L}_{\mathrm{FM}}(\theta; \boldsymbol{c}_\theta) + \nabla_\theta R(\theta). \tag{14}$$

Thus, for non-stopped $\boldsymbol{c}_\theta$, CFM and FM are not gradient-equivalent in general. If $\bar{\boldsymbol{c}}_\theta = \mathbf{sg}(\boldsymbol{c}_\theta)$, then R is fixed in the FM/CFM computation graph, so the extra term vanishes and the standard FM/CFM gradient equivalence is recovered. In our architecture, this is precisely why we apply stop-gradient to the shared embedding before using it to condition the generative and prior branches.

It is worth naming what the extra term rewards, and what that would do to the representation. By the tower property $R(\theta) = \mathbb{E}\big[\mathrm{Var}(\boldsymbol{v}_t \mid \boldsymbol{x}_t, t, \boldsymbol{c})\big]$, where Var is the total conditional variance. From $\boldsymbol{x}_t = t\boldsymbol{x} + (1-t)\boldsymbol{\epsilon}$ we have $\boldsymbol{x} - \boldsymbol{x}_t = (1-t)(\boldsymbol{x} - \boldsymbol{\epsilon})$, so the velocity target is the affine function $\boldsymbol{v}_t = (\boldsymbol{x} - \boldsymbol{x}_t)/(1-t)$ of the clean sample, whose multiplier and offset are both fixed once $(\boldsymbol{x}_t, t)$ are given. Its variance therefore scales by $(1-t)^{-2}$, and

$$R(\theta) = \mathbb{E}_{\boldsymbol{x}_t,t,\boldsymbol{c}}\left[\frac{\mathrm{Var}(\boldsymbol{x} \mid \boldsymbol{x}_t, t, \boldsymbol{c}_\theta(\boldsymbol{x}))}{(1-t)^2}\right], \tag{15}$$

is the residual uncertainty about the *clean image* once the noised view and the conditioning are known. Here θ enters only through $\boldsymbol{c}_\theta(\boldsymbol{x})$, since $\boldsymbol{v}_t$, $\boldsymbol{x}_t$ and t do not involve it. The two extremes bound what that term can do: an uninformative $\boldsymbol{c}$ leaves the full posterior uncertainty $\mathrm{Var}(\boldsymbol{x} \mid \boldsymbol{x}_t, t)$, while a $\boldsymbol{c}$ that determines $\boldsymbol{x}$ drives R to zero. Lowering R therefore means making $\boldsymbol{c}_\theta(\boldsymbol{x})$ more informative about $\boldsymbol{x}$ itself.

What matters for the representation is which information that recruits. The cheapest way to pin down $\boldsymbol{x}$ from $\boldsymbol{x}_t$ is to carry the detail that noising destroys, and in the limit R is minimized by a $\boldsymbol{c}$ that is a lossless code of $\boldsymbol{x}$, at which point the embedding is the image again and carries no abstraction. That is exactly the information $\mathcal{L}_{\mathrm{CL}}$ is built to discard, since augmentation invariance means being blind to crops, color, grayscale, and other augmentations. Without the stop-gradient, the generative objective would therefore not merely break an identity: it would affect the contents of the shared embedding, and the solution it prefers is a compression code rather than a representation. The stop-gradient does not prevent generation from benefiting from whatever $\boldsymbol{e}$ carries, it prevents the generative loss from choosing what that is, leaving that to $\mathcal{L}_{\mathrm{CL}}$.

Proof *Following Theorem 2 of Lipman et al. (2023), write $V_\theta = \boldsymbol{v}_\theta(\boldsymbol{x}_t, t \mid \boldsymbol{c})$ and $m_t = \boldsymbol{v}_t^\star(\boldsymbol{x}_t \mid \boldsymbol{c})$. Expanding the two squared losses gives*

$$\|V_\theta - \boldsymbol{v}_t\|^2 = \|V_\theta\|^2 - 2\langle V_\theta, \boldsymbol{v}_t\rangle + \|\boldsymbol{v}_t\|^2, \tag{16}$$

$$\|V_\theta - m_t\|^2 = \|V_\theta\|^2 - 2\langle V_\theta, m_t\rangle + \|m_t\|^2. \tag{17}$$

The first terms are identical. For the cross term, by definition

$$m_t = \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon}}[\boldsymbol{v}_t \mid \boldsymbol{x}_t, t, \boldsymbol{c}],$$

and $V_\theta = \boldsymbol{v}_\theta(\boldsymbol{x}_t, t \mid \boldsymbol{c})$ is a deterministic function of the same conditioning variables. The first equality below is then the tower property, using that $(\boldsymbol{x}_t, t, \boldsymbol{c})$ is a function of $(\boldsymbol{x}, \boldsymbol{\epsilon}, t)$; the second takes V_θ inside the inner conditional expectation, which is legitimate because it is measurable with respect to exactly the variables being conditioned on; the third is the tower property again. This is the conditioning step of the Flow Matching proof with the conditioning set enlarged to include $\boldsymbol{c}$,

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},t}[\langle V_\theta, m_t\rangle] &= \mathbb{E}_{\boldsymbol{x}_t,t,\boldsymbol{c}}[\langle V_\theta, \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon}}[\boldsymbol{v}_t \mid \boldsymbol{x}_t, t, \boldsymbol{c}]\rangle] \\ &= \mathbb{E}_{\boldsymbol{x}_t,t,\boldsymbol{c}}[\mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon}}[\langle V_\theta, \boldsymbol{v}_t\rangle \mid \boldsymbol{x}_t, t, \boldsymbol{c}]] \\ &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},t}[\langle V_\theta, \boldsymbol{v}_t\rangle]\,. \end{aligned} \tag{18}$$

None of those three steps requires $\boldsymbol{c}$ to be independent of θ, so a learned conditioning leaves the cross term intact and its entire effect is confined to the last terms. Those are not equal in general: the CFM objective contains $\mathbb{E}\|\boldsymbol{v}_t\|^2$, whereas the FM objective contains $\mathbb{E}\|m_t\|^2$. Instead, the difference is the expected residual variance of the target velocity after conditioning on $(\boldsymbol{x}_t, t, \boldsymbol{c})$:

$$\begin{aligned}\mathcal{L}_{\mathrm{CFM}}(\theta; \boldsymbol{c}_\theta) - \mathcal{L}_{\mathrm{FM}}(\theta; \boldsymbol{c}_\theta) &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},t}\|\boldsymbol{v}_t\|^2 - \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},t}\|m_t\|^2 \\ &= \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},t}\left[\|\boldsymbol{v}_t - m_t\|^2\right]. \end{aligned} \tag{19}$$

The last equality is the conditioning step once more, with m_t now playing the role V_θ played above. Since m_t is itself a function of $(\boldsymbol{x}_t, t, \boldsymbol{c})$,

$$\mathbb{E}\langle \boldsymbol{v}_t, m_t\rangle = \mathbb{E}\big[\langle \mathbb{E}[\boldsymbol{v}_t \mid \boldsymbol{x}_t, t, \boldsymbol{c}],\, m_t\rangle\big] = \mathbb{E}\|m_t\|^2,$$

so expanding $\mathbb{E}\|\boldsymbol{v}_t - m_t\|^2 = \mathbb{E}\|\boldsymbol{v}_t\|^2 - 2\mathbb{E}\langle \boldsymbol{v}_t, m_t\rangle + \mathbb{E}\|m_t\|^2$ collapses the cross term and leaves $\mathbb{E}\|\boldsymbol{v}_t\|^2 - \mathbb{E}\|m_t\|^2$.

Therefore, when the conditioning value is treated as fixed, this difference is independent of the velocity model parameters, and the CFM and FM gradients with respect to those parameters are equal. However, when $\boldsymbol{c}_\theta$ is not fixed, m_t and the conditional variance term $R(\theta)$ both depend on θ through the conditioning path. Differentiating Equation 13 gives Equation 14. Since $\nabla_\theta R(\theta)$ need not be zero, the gradients are not equivalent in general. With stop-gradient, $\partial \bar{\boldsymbol{c}}_\theta / \partial\theta = 0$, so the derivative of R through the conditioning path is zero and the standard fixed-conditioning FM identity applies.

Short counterexample without stop-gradient. The failure can already occur for a deterministic linear encoder. Take $\boldsymbol{x}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathrm{diag}(2, 1))$ independently in $\mathbb{R}^2$ and fix $t = \frac{1}{2}$, so that $\boldsymbol{x}_t = \frac{1}{2}(\boldsymbol{x} + \boldsymbol{\epsilon})$ and $\boldsymbol{v}_t = \boldsymbol{x} - \boldsymbol{\epsilon}$, and let the bottleneck encoder be $\boldsymbol{c}_\theta(\boldsymbol{x}) = x_1 + \theta x_2$.

Since $\mathrm{Cov}(\boldsymbol{v}_t, \boldsymbol{x}_t) = \frac{1}{2}\,\mathrm{diag}(2, 1) - \frac{1}{2}\,\mathrm{diag}(2, 1) = 0$ and the pair is jointly Gaussian, $\boldsymbol{v}_t$ is independent of $\boldsymbol{x}_t$, so whatever $(\boldsymbol{x}_t, \boldsymbol{c}_\theta)$ reveals about $\boldsymbol{v}_t$ arrives through $\boldsymbol{c}_\theta$. Substituting the definitions gives $\boldsymbol{c}_\theta - x_{t,1} - \theta x_{t,2} = \frac{1}{2}(v_{t,1} + \theta v_{t,2})$, so conditioning on $(\boldsymbol{x}_t, \boldsymbol{c}_\theta)$ is conditioning on the single statistic $U = v_{t,1} + \theta v_{t,2}$, and $m_t = \mathbb{E}[\boldsymbol{v}_t \mid U]$.

Now $\boldsymbol{v}_t \sim \mathcal{N}(0, \Sigma_v)$ with $\Sigma_v = \mathrm{diag}(4, 2)$, so $\mathrm{Cov}(\boldsymbol{v}_t, U) = \Sigma_v(1, \theta)^\top = (4, 2\theta)^\top$ and $\mathrm{Var}(U) = 4 + 2\theta^2$. For a jointly Gaussian pair the conditional mean is the linear projection $m_t = \mathrm{Cov}(\boldsymbol{v}_t, U)\, U / \mathrm{Var}(U)$, whose residual has covariance $\Sigma_v - \mathrm{Cov}(\boldsymbol{v}_t, U)\,\mathrm{Cov}(\boldsymbol{v}_t, U)^\top / \mathrm{Var}(U)$. Taking the trace of that residual covariance,

$$R(\theta) = \mathrm{tr}\,\Sigma_v - \frac{\|\,\mathrm{Cov}(\boldsymbol{v}_t, U)\|^2}{\mathrm{Var}(U)} = 6 - \frac{16 + 4\theta^2}{4 + 2\theta^2} = \frac{4 + 4\theta^2}{2 + \theta^2}, \qquad \nabla_\theta R(\theta) = \frac{8\theta}{(2 + \theta^2)^2},$$

which is nonzero for $\theta \neq 0$. The residual variance therefore does depend on the learned conditioning, and without stop-gradient the FM/CFM gradient equivalence fails.

The example also shows the *direction* of this dependence. R is minimized at $\theta = 0$, where $c_\theta = x_1$ and $U = v_{t,1}$, the highest-variance coordinate of v_t. Thus, in this toy problem, the additional FM term favors conditioning directions that explain more raw target variance. For images, this motivates the concern that, unless the contrastive objective explicitly constrains the representation away from such directions, the FM term can favor low-level variance that the representation objective is intended to discard.

E EXPERIMENT DETAILS

This appendix gives the training and evaluation setup behind Section 4. Table 12 summarizes the main hyperparameters at both scales. Unless an experiment states otherwise, the settings below are the defaults for every ImageNet-1K (Deng et al., 2009; Russakovsky et al., 2015) run.

All experiments are run on nodes of 8 NVIDIA H200 GPUs. To speed up experiments, we train on up to 3 nodes for B/16 and 6 nodes for L/16.

E.1 TRAINING SETUP

We follow JiT (Li & He, 2026) for the generative hyperparameters and for $\mathcal{L}_{\mathrm{CFM}}$, including the optimizer, the augmentation policy of the original image and the flow matching loss. We depart from it in two places, both chosen once and applied to every image generator in the comparison, baselines included: the base learning rate, which is tuned to $1 \cdot 10^{-4}$; and the attention residuals of Appendix F. Since every arm trains at a global batch of 1024, the learning rate is $4 \cdot 10^{-4}$. For the contrastive hyperparameters and the augmentation policy of the augmented images we follow SogCLR (Yuan et al., 2022).

The RCG baseline is our own re-implementation on this backbone and conditions on DINOv2 (Oquab et al., 2024) embeddings. For its prior, we use the architecture of their official implementation[1], a residual MLP with 12 blocks of width 1536, trained for 200 epochs with our flow-matching objective and prior timestep schedule. For a fair comparison, we also ran RCG with our prior, which performed worse.

Augmentations. We use $k_O = 1$ original image and $k_A = 2$ augmented images. The original image follows JiT's preprocessing, a center crop to the training resolution and a random horizontal flip with probability 0.5. It is the only input the generative path sees and it is also one contrastive anchor. The two augmented images are used by the representation path alone and follow the SimCLR policy (Chen et al., 2020a; Grill et al., 2020): random resized crop with scale in $[0.08, 1.0]$, color jitter of strength $(0.4, 0.4, 0.2, 0.1)$ applied with probability 0.8, random grayscale with probability 0.2, Gaussian blur with σ uniform in $[0.1, 2.0]$, and a random horizontal flip. This is what exposes the shared encoder to both regimes every step, noised original images for generation and clean augmented ones for the invariances.

Timestep sampling. For both flow matching losses we sample t from the logit-normal schedule (Esser et al., 2024): draw $s \sim \mathcal{N}(\mu, \sigma^2)$ and set $t = \mathrm{sigmoid}(s)$, which we write $\mathrm{logit}(t) \sim \mathcal{N}(\mu, \sigma^2)$. The image path uses JiT's default $(\mu, \sigma) = (-0.8, 0.8)$, and the prior path $(\mu', \sigma') = (-1.2, 0.6)$, the latter selected on a frozen-prior grid and applied to the RCG baseline as well.

Losses. The contrastive objective is the multi-view form of the Global Contrastive Loss (Yuan et al., 2022), with fixed temperature $\tau = 0.15$, moving-average parameter $\gamma = 0.9$, and negatives gathered across GPUs. The generative objective is the conditional flow matching loss (Lipman et al., 2023) with $\boldsymbol{x}$-prediction and the velocity parameterization of JiT (Li & He, 2026). We clamp the $1 - t$ denominator at $\max(1 - t, 0.05)$ for numerical stability near $t = 1$. The weight λ on $\mathcal{L}_{\mathrm{CL}}$ comes from the balancer of Section 3.3 and Algorithm 2: gradient norms are read on the feed-forward submodule of the last encoder block, the estimate is refreshed every 10 steps, and the two moving averages use a decay ramped linearly from 0.9 to 0.99 over the first 5000 steps. Their ratio is clamped to $[10^{-3}, 1]$ and multiplied by $\lambda_0 = 1$. The CBC weight is fixed at $\eta = 0.01$ and is not balanced. *None of these constants are tuned.*

[1] `https://github.com/LTH14/rcg`

Table 12: **Main hyperparameters** for the reported SCION (B/16) and SCION (L/16) ImageNet-1K 256×256 runs. Rows spanning both columns are identical at the two scales. The baselines share this recipe apart from the objective and, for REPA and iREPA, the $4/8$ encoder–decoder split, following their published split; see Appendix E. Guidance follows Ho & Salimans (2021) on the interval of Kynkäänniemi et al. (2024).

	SCION (B/16)	**SCION (L/16)**
Architecture		
Depth: total / enc. / dec.	12, 8, 4	24, 16, 8
Hidden dim	768	1024
Heads	12	16
In-context start block	4	8
`[CLS]` normalization	identity	BatchNorm
Image size	256	
Patch size	image size / 16	
Patch bottleneck	128	
In-context tokens	32	
Training		
Epochs	400 (800 system-level)	400
Optimizer	AdamW	Muon (block matrices) + AdamW
Learning rate	$4 \cdot 10^{-4}$	$2 \cdot 10^{-2}$ Muon, $4 \cdot 10^{-4}$ AdamW
Weight decay	0.0	$5 \cdot 10^{-4}$ Muon, 0.0 AdamW
LR schedule	constant	
Warmup epochs	5	
AdamW (β_1, β_2)	(0.9, 0.95)	
Global batch	1024	
Conditioning drop prob.	10%	
Number augmented views	2	
EMA decays	0.9999 (EMA1), 0.9996 (EMA2)	
$\mathrm{logit}(t)$ schedule	$\mathcal{N}(-0.8, 0.8^2)$	
Prior $\mathrm{logit}(t)$ schedule	$\mathcal{N}(-1.2, 0.6^2)$	
CL dimension	256	
CL temperature / γ	0.15, 0.9	
CBC weight η	0.01	
Gradient-ratio λ_0	1	
Gradient-ratio update	every 10 steps	
Gradient-ratio EMA warmup	0.9 to 0.99 over 5000 steps	
Gradient-ratio clamp	$[10^{-3}, 1]$	
t-division clamp	0.05	
Sampling		
ODE solver / steps	Heun, 50	
Sampling t schedule	linear in $[0.0, 1.0]$	
CFG interval	$[0.1, 1.0]$	
CFG sweep start / step	from 1.0, step 0.1, extended while FID improves	
CFG sweep images	8000	

Optimization. At B/16 we train with AdamW (Loshchilov & Hutter, 2019), $(\beta_1, \beta_2) = (0.9, 0.95)$, in bfloat16 mixed precision, under a constant learning rate with a linear warmup over 5 epochs and weight decay 0. At L/16 the parameters split across two optimizers: Muon (Jordan et al., 2024) takes the two-dimensional parameters inside the transformer blocks, at learning rate $2 \cdot 10^{-2}$ and weight decay $5 \cdot 10^{-4}$, and everything else stays on AdamW at the same learning rate and weight decay as at B/16. Moreover, for L/16, we use a BatchNorm (Ioffe & Szegedy, 2015) after the `[CLS]` token at the encoder-decoder split. We keep two exponential moving averages of the weights, with decays 0.9999 and 0.9996.

E.2 Evaluation

Checkpoint selection. All metrics are computed on EMA weights. Section 4.1 reports the 400-epoch checkpoint, the budget at which every baseline stops, Section 4.3 reports 200 epochs, and Section 4.2 reports SCION (B/16) at 800 epochs and SCION (L/16) at 400. We evaluate the EMA

with decay 0.9999, except for the system-level comparison, where we evaluate both exponential moving averages and select the 0.9996 one only for SCION (L/16).

Generation. All generation numbers use 50K samples drawn with a Heun ODE solver (Heun, 1900) with 50 steps (49 Heun steps and a final Euler step). For the class-conditional baselines the batch is 50 samples for each of the 1000 ImageNet classes. SCION is class-unconditional and does not use labels for generation either. Instead, we unconditionally sample from the learned prior e_ω the conditioning vectors for the generator. Thus, its batch carries no class structure by construction. Where guidance is reported, it is applied on the interval $t \in [0.1, 1.0]$ (Kynkäänniemi et al., 2024) at a scale swept from 1.0 in steps of 0.1 on 8000 samples, continuing while FID keeps improving.

Generation metrics. We evaluate generation with the ADM suite (Dhariwal & Nichol, 2021), using their reference batches and official implementation.[2] All metrics are computed from Inception-V3 outputs on the 50K generated images; all but the Inception Score also read the suite's ImageNet 256×256 reference batch.

- *FID* (Heusel et al., 2017) is the Fréchet distance between Gaussians fitted to the pooled Inception features of the generated and reference sets.
- *sFID* (Nash et al., 2021) computes the same distance on flattened intermediate *spatial* features instead of pooled ones, making it more sensitive to errors in spatial structure.
- *IS* (Salimans et al., 2016) is the exponential of the average KL divergence between the class distribution the same Inception-V3 predicts for each generated image and the marginal over all of them. It rises when samples are individually classifiable and collectively spread across classes, and it is the one metric here computed from the generated set alone.
- *Precision* and *Recall* (Kynkäänniemi et al., 2019) estimate the support of each distribution with a k-nearest-neighbor manifold ($k = 3$). Precision is the fraction of *generated* images inside the reference manifold and measures fidelity. Recall is the fraction of *reference* images inside the generated manifold and measures coverage. Since guidance typically buys fidelity at the cost of coverage, the two should be read together.

Not all metrics see the same reference set. FID and sFID are computed against first- and second-moment statistics precomputed over the full reference dataset, while precision and recall are estimated against the $10K$ reference images distributed with the batch, not against the full dataset.

Representation. Both probes freeze the network and read one vector per image. For SCION that vector is the `[CLS]` token at the encoder/decoder split, taken before the contrastive projection head. The baselines carry no `[CLS]` token, so for them it is the mean of the patch tokens, excluding the in-context conditioning tokens, at their own split. Vanilla JiT and the RCG generator have no split and are read after their last block. The contrastive-only controls carry a `[CLS]` token and are read like SCION.

The ImageNet probe (*Acc.*) fits an ℓ_2-regularized multinomial logistic regression on the raw features, with no augmentation: images are resized to 292 with bicubic interpolation and center-cropped to 256. The regularization strength is chosen over 45 logarithmically spaced values between 10^{-6} and 10^5 on a held-out 10% of the training split, the classifier is then refit on the full training split at that value, and we report top-1 accuracy on the validation split.

The transfer probe (*transfer-8*) evaluates out-of-domain transfer on Food-101 (Bossard et al., 2014), CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009), Stanford Cars (Krause et al., 2013), Describable Textures (Cimpoi et al., 2014), Oxford-IIIT Pets (Parkhi et al., 2012), Caltech-101 (Li et al., 2022) and Oxford 102 Flowers (Nilsback & Zisserman, 2008), and reports their unweighted mean. We follow the evaluation protocol of each dataset's own paper, so mean per-class accuracy for Caltech-101, Oxford-IIIT Pets and Oxford 102 Flowers and top-1 accuracy for the rest. Features are ℓ_2-normalized and standardized. We then fit an ℓ_2-regularized multinomial logistic regression with the L-BFGS solver of scikit-learn (Pedregosa et al., 2011). Training images are center-cropped to the training resolution of 256 with a random horizontal flip, and the flip is removed at test time. We

[2] `https://github.com/openai/guided-diffusion/tree/main/evaluations`

report the best test accuracy over 10 logarithmically spaced regularization values between 10^{-3} and 1.

F ARCHITECTURE DETAILS

SCION is a JiT-style pixel-space transformer (Li & He, 2026) with three changes. First, we split the transformer depth into a shared encoder f_θ and a generative decoder g_ϕ, two contiguous portions of one ViT-style backbone (Dosovitskiy et al., 2021) with total depth $L = L_{\text{enc}} + L_{\text{dec}}$. Second, we prepend a learnable `[CLS]` token and read it out at the encoder/decoder boundary as the global embedding $\boldsymbol{e}$. Third, we attach a contrastive projection head h_ψ to that embedding while the patch tokens continue through the decoder for flow matching. Additionally, we condition the generative path on $\mathbf{sg}(\boldsymbol{e})$ and learn a prior e_ω (Li et al., 2024a) over these embeddings for class-unconditional sampling. Table 13 lists each component with its input and output spaces.

All remaining architectural choices follow JiT, except for the attention residuals below, which every model we train uses. For completeness, we go into each component in the following paragraphs.

Transformer block. Each block uses pre-RMSNORM (Zhang & Sennrich, 2019), multi-head self-attention with QK-norm (Henry et al., 2020) and two-dimensional rotary positional embeddings (Su et al., 2024), then a second pre-RMSNORM and a SWIGLU feed-forward network (Shazeer, 2020) whose inner width is $\frac{8}{3}d$, an MLP ratio of 4 under SWIGLU's two-thirds rescaling. As in DiT and JiT, both residual branches are modulated by adaLN-Zero (Peebles & Xie, 2023): a linear projection of the conditioning vector produces shift, scale and gate parameters for the two sublayers, and the modulation layers are zero-initialized so each branch begins near the identity.

Attention residuals. Applied to every model we train in Section 4, ours and all baselines, attention residuals (Kimi Team et al., 2026) replace the plain residual add of each sublayer. The sublayers stay pre-norm, since what changes is the residual stream and not where the normalization sits. Instead of reading the running residual sum, a sublayer reads a learned softmax-weighted mixture over its depth history and appends its own output to that history: the keys are the normalized history entries, and the query is a learned vector per token role (patch, `[CLS]` or in-context tokens). Writing $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n \in \mathbb{R}^d$ for the entries at one token position of role ρ, that mixture is

$$\hat{\boldsymbol{s}} \;=\; \sum_{i=1}^{n} \alpha_i \, \boldsymbol{s}_i, \qquad \alpha_i \;=\; \frac{\exp \langle \boldsymbol{q}_\rho, \, \mathrm{RMSNorm}(\boldsymbol{s}_i) \rangle}{\sum_{l=1}^{n} \exp \langle \boldsymbol{q}_\rho, \, \mathrm{RMSNorm}(\boldsymbol{s}_l) \rangle}, \tag{20}$$

a softmax over depth with no $1/\sqrt{d}$ scaling. A history entry is the sum of four consecutive sublayer outputs, which bounds the number of entries the mixture runs over. The queries are zero-initialized, so the mixture is uniform at initialization.

Patch and positional embeddings. Images are tokenized by a low-rank linear patch embedder (Li & He, 2026): a $p \times p$ strided convolution maps non-overlapping RGB patches into a bottleneck of width 128, followed by a 1×1 convolution to the transformer width d. Two positional signals are used and they act at different places. Fixed two-dimensional sine-cosine embeddings (Vaswani et al., 2017) are added to the patch tokens only, before the `[CLS]` and in-context tokens are concatenated. The rotary embeddings act inside attention, on the queries and keys after QK-norm, with an identity rotation at the non-patch positions so that the `[CLS]` and in-context tokens carry no spatial position. The final layer applies RMSNORM, adaLN modulation and a zero-initialized linear projection to $p^2 c_{\text{out}}$ values per patch, which are unpatchified into an image.

Conditioning. The timestep is embedded with sinusoidal features followed by a two-layer SILU (Elfwing et al., 2018) MLP. The conditioning vector $\boldsymbol{c}$ passes through a linear projection and then enters the network twice. It is added to the timestep embedding that drives the adaLN-Zero modulation of every block, and the same projected vector is also injected as 32 in-context tokens at a fixed block, following the design of JiT and MAR (Li & He, 2026; Li et al., 2024b). These tokens are added on the fourth block for B/16 and the eighth for L/16. The in-context tokens are removed before the final image projection. On the representation pass the conditioning input is the learned null $\boldsymbol{e}_\varnothing$, and classifier-free guidance uses a second learned null $\boldsymbol{c}_\varnothing$.

Encoder/decoder split. At block L_{enc} the `[CLS]` token is read out as the global embedding. At B/16 the readout is the raw token; at L/16 it passes through a BATCHNORM (Ioffe & Szegedy, 2015). The `[CLS]` token is dropped at the split and never reaches the decoder, so the decoder is a function of the patch tokens $\boldsymbol{h}$ alone. Our SCION (B/16) splits $(L_{\text{enc}}, L_{\text{dec}}) = (8, 4)$ of 12 blocks at width 768 with 12 heads, and SCION (L/16) splits $(16, 8)$ of 24 at width 1024 with 16 heads.

Contrastive head. The projection head h_ψ is a two-hidden-layer MLP, $d \to 2048 \to 2048 \to 256$, followed by ℓ_2-normalization. It is used by $\mathcal{L}_{\text{CL}}$ and by nothing else, so it is discarded after training.

Prior. It is a residual MLP over a single vector with additive timestep conditioning. That is, a linear layer maps the noised embedding $\widetilde{\boldsymbol{e}}_{t'} \in \mathbb{R}^d$ to width C. The timestep is encoded with 256 sinusoidal features and a two-layer SILU MLP to width C. The hidden vector then passes through N residual blocks, each applying RMSNORM, a SWIGLU projection to the block width, an additive linear projection of the timestep embedding, and a linear projection back to C. A final RMSNORM, SILU and linear layer return to $\mathbb{R}^d$. Every block's output projection and the final projection are zero-initialized, so each block is the identity and the prior predicts zero at initialization. The prior uses additive residual connections and it is trained with the same $\boldsymbol{x}$-prediction flow matching parameterization as the image path, in representation space. Table 14 gives the configuration.

Table 13: **Components of the shared model.** T is the number of patch tokens, d the hidden width, d_z the contrastive projection dimension, and $\boldsymbol{c}$ the conditioning vector of Section 3.1.

Component	Symbol	Input	Output
Shared encoder	f_θ	$(\boldsymbol{x}_t \mid t, \boldsymbol{c}) \in \mathbb{R}^{3\times H\times W} \times [0,1] \times \mathbb{R}^d$	$(\boldsymbol{e}, \boldsymbol{h}) \in \mathbb{R}^d \times \mathbb{R}^{T\times d}$
Noisy image, timestep and conditioning $\mapsto$ global embedding $\boldsymbol{e}$ and patch tokens $\boldsymbol{h}$.			
Generative decoder	g_ϕ	$(\boldsymbol{h} \mid t, \boldsymbol{c}) \in \mathbb{R}^{T\times d} \times [0,1] \times \mathbb{R}^d$	$\boldsymbol{x} \in \mathbb{R}^{3\times H\times W}$
Patch tokens, timestep and conditioning $\mapsto$ predicted clean image.			
Contrastive projection	h_ψ	$\boldsymbol{e} \in \mathbb{R}^d$	$\boldsymbol{z} \in \mathbb{R}^{d_z}$
Global embedding $\mapsto$ the space where $\mathcal{L}_{\text{CL}}$ is optimized.			
Embedding prior	e_ω	$(\widetilde{\boldsymbol{e}}_{t'}, t') \in \mathbb{R}^d \times [0,1]$	$\bar{\boldsymbol{e}} \in \mathbb{R}^d$
Noised standardized embedding $\mapsto$ its clean standardized counterpart.			

Table 14: **Prior architecture.** The residual-MLP backbone of e_ω, shared by both scales; its input and output width follows the backbone, so the parameter count is the B/16 one.

Config	Value
Input / output width	representation dimension d
Hidden width C	808
Residual blocks N	14
Block SWIGLU width	2155
Timestep features	sinusoidal, width 256
Timestep embedding	808, added inside each block
Activation	SILU
Normalization	RMSNORM
Output head	RMSNORM, SILU, linear
Zero-initialized	block and final output projections
Dropout	0.0
Parameters (B/16)	99.7M

G QUALITATIVE RESULTS USING LEARNED REPRESENTATIONS

SCION extracts its own semantic representations and uses them to condition generation. We explore this connection through two qualitative experiments, both using representations produced by the model itself. First, we generate from real-image representations to examine whether the outputs preserve the references' semantic content. We then interpolate between two image representations to explore how the generated content changes along the resulting path.

All examples use SCION (L/16) at 400 epochs. Generated images use representation guidance with a scaling factor $s = 2.0$ (see Equation 6). Reference images come from the ImageNet training set and use the same 256×256 crop supplied to the encoder. This section is meant to illustrate qualitative behavior rather than quantify reconstruction accuracy or diversity.

G.1 GENERATION FROM REAL-IMAGE REPRESENTATIONS

We encode each reference image and directly supply its representation to the generator, bypassing the prior. The pairs in Figure 4 illustrate preservation of semantic content without exact reproduction of image details. Generated images retain recognizable subjects and scene types, while pose, color, object markings, backgrounds, and architectural details can change. For example, a cup retains its identity as a cup while its decoration changes, and a building remains recognizable as the same type of structure despite changes in its facade. Thus, the model's own representations support semantically related generation without requiring the output to reconstruct the particular reference image.

G.2 INTERPOLATION BETWEEN IMAGE REPRESENTATIONS

We next interpolate between the representations of two real images and generate from the intermediate vectors (Figure 5). We use spherical interpolation of their directions and linear interpolation of their magnitudes in the generator's conditioning space, bypassing the prior. Image-sampling noise is held fixed within each row, so changes along a path reflect changes in the conditioning representation. The paths first explore appearance and scene composition within a class, then connect different classes, from two dog breeds to more distant concepts such as a bottle and a dog. These cross-class paths reveal how structures from the endpoints can blend, although their intermediate images need not depict plausible natural objects.

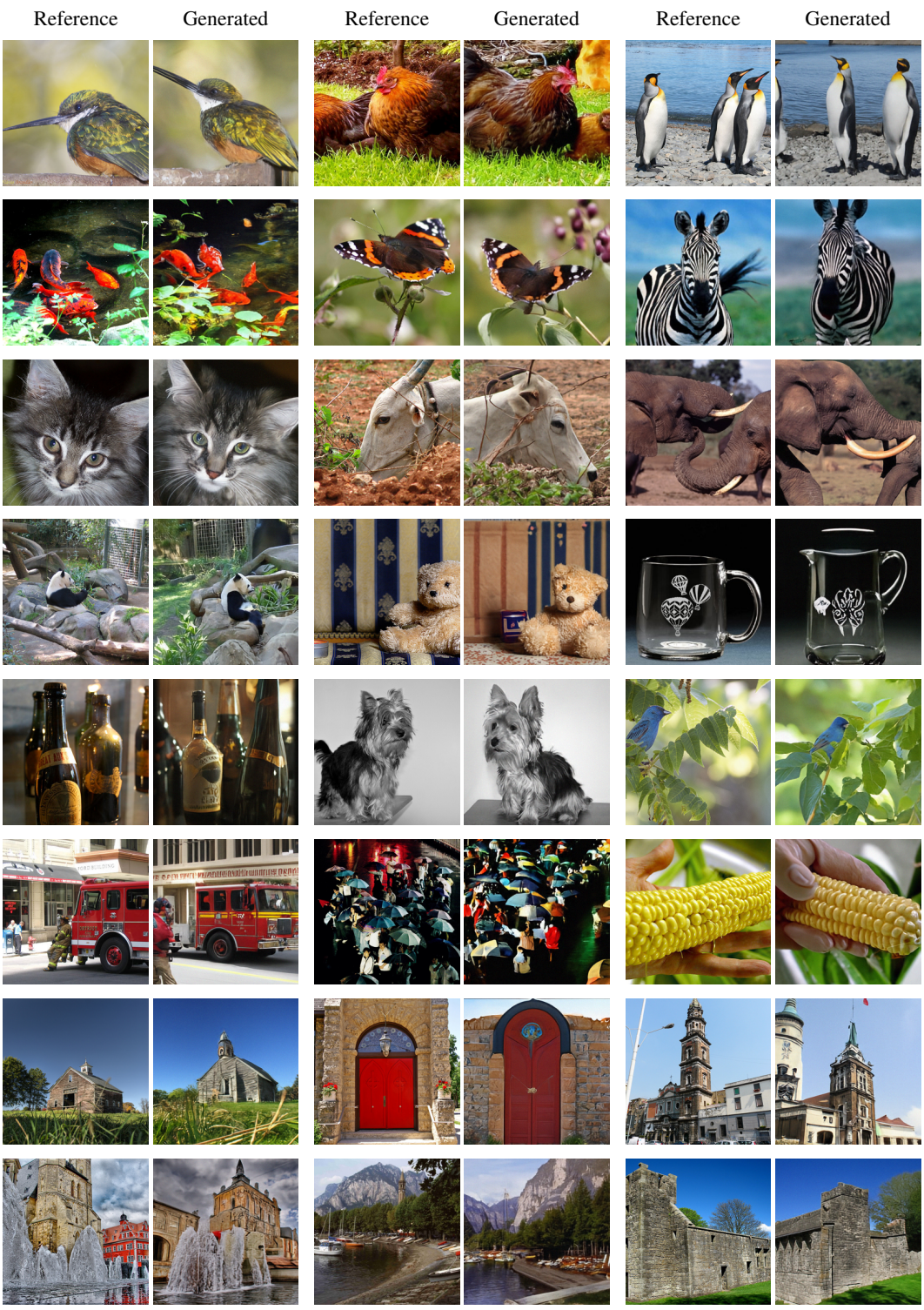

Figure 4: **Preserving semantics while varying image details.** Each pair shows a real reference and a generation conditioned on the representation extracted by SCION itself. Recognizable subjects and scene types persist across changes in pose, color, markings, background, and structure. The generator starts from image noise and bypasses the learned prior. All examples use SCION (JiT-L/16) at 400 epochs with representation guidance.

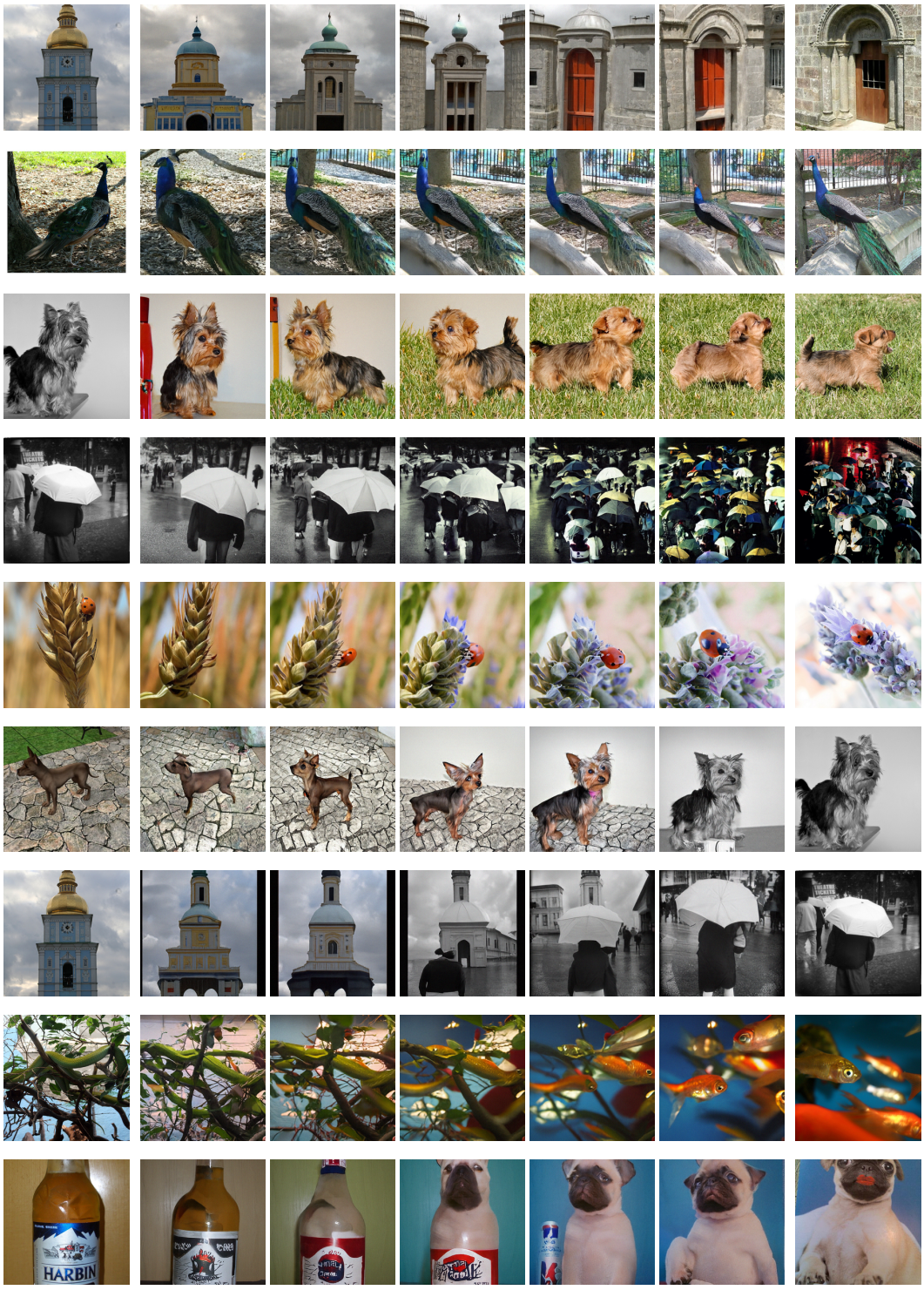

Figure 5: **Interpolation in the learned conditioning space.** Real references appear at the outer edges. All five interior images are generated from interpolated representations, with $\alpha = 0$ and 1 using the endpoint representations directly. The first five rows connect images from the same ImageNet class, while the last four connect different classes. Image noise is fixed within each row. All paths use spherical interpolation, and SCION (JiT-L/16) at 400 epochs with representation guidance.