IMAGE TOKENIZERS SHOULD BE TRAINED BETWEEN IMAGES

Anonymous authors
Paper under double-blind review

ABSTRACT

A visual tokenizer encodes an image into a smaller representation, and its decoder turns that code back into the image. However, the decoder is trained only at the codes the encoder produces from the training images. It never learns what to do with codes that lie in the space between training samples. Codes in this space lie outside of the training distribution and the decoder produces blurry images with strong artifacts there. For image generation, a second model, the generator, learns to produce images as codes of the encoder of a tokenizer. But, if generated codes land outside of what the decoder expects, the decoded image will contain imperfections. In order to make the decoder more robust to generated codes, current methods add noise to codes during decoder training. However, recent research indicates that generated codes do not simply contain noise. Instead, they lie between the codes the generator was trained on, and resemble an average of codes of training images more than one of them plus noise. Therefore, to improve quality of generated images, decoders should be trained to handle codes in the space between training samples. Motivated by these insights, we propose **InterpTok**, which fine-tunes MAETok by training the decoder to generate natural images on mixtures of the codes the encoder produces. But no target image exists for an image between two training samples. So a discriminator judges whether each decoded mixture of codes looks as natural as the images whose codes were mixed. Additionally, a mixing ratio loss requires the decoded image to visually keep the mixing ratio of the two codes. This prevents a collapse to a trivial solution and trains the decoder to produce a sharp and natural composition of both images. InterpTok keeps MAETok's reconstruction quality and improves interpolation quality more than threefold, the best among the thirteen tokenizers of a released comparison. Beyond interpolation, we show that our method can substantially improve generation training speed as well as the generation quality with much less computational cost. InterpTok improves the generation FID of a matched diffusion transformer from 11.61 to 8.48 without guidance, and even reaches 8.11 at an eighth of the sampling steps, where the generator on MAETok's codes worsens. Furthermore, the generator trained on our tokenizer surpasses the final results of the generator trained on MAETok at half the training steps. Our findings show that the space between image codes can and should be explicitly supervised rather than ignored.

1 INTRODUCTION

Modern image generation largely relies on diffusion and flow matching models (Ho et al., 2020; Ma et al., 2024). These models generate images through many refinement steps, so most operate in a compact latent space to make each step more efficient (Rombach et al., 2022; Peebles & Xie, 2023). In this space, each image is represented by a compact code created by an image tokenizer. The tokenizer's encoder turns images into codes, and its decoder turns codes back into images. The generator learns to produce these codes and therefore depends on the space created by the tokenizer.

Improving image generation therefore starts with asking what makes this latent space useful for a generator. Most tokenizer methods focus on the codes of the training images and make them easier for a generator to learn (Chen et al., 2025a; Yao et al., 2025; Fan et al., 2026). Standard tokenizer training, however, supervises one code at a time by reconstructing its image. It tells the decoder what to produce at the codes of the training images, but says nothing about the codes between them. As

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

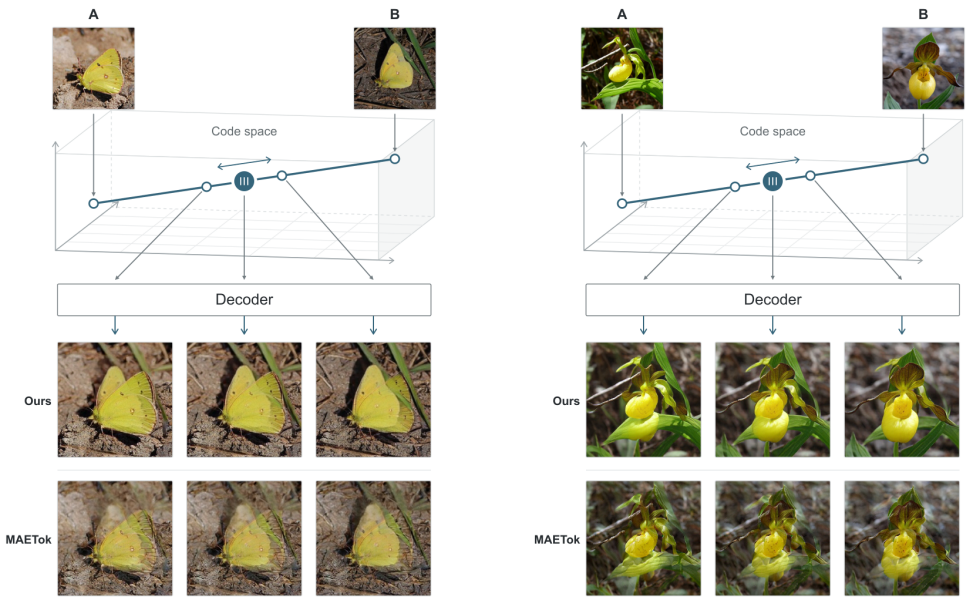


Figure 1: **Overview.** Reconstruction trains the tokenizer at the codes of images A and B, never on the line between them. InterpTok trains it there, and the marked codes decode to images as natural as A and B, changing from A toward B.

Fig. 1 illustrates, a decoder can reconstruct images A and B well and still turn a mixture of their codes into a blurry superposition. Reconstruction can therefore give us good endpoints while leaving the path between them undefined.

This gap matters because a generator can produce codes that the tokenizer never saw during training. Recent work shows that generated codes lie near the lines between neighboring training codes. It also shows that the quality of images decoded on these lines predicts generation quality better than reconstruction quality (Xu et al., 2026). The decoder must therefore work beyond the exact codes it was trained to reconstruct. Existing tokenizer and decoder training schemes add noise to training codes and make their local neighborhoods more robust (Yang et al., 2026b; Zheng et al., 2026). They improve the visual quality of decoded codes in the space around each image code, but they do not define what should happen along the line from one image code to another.

This observation leads to the central idea of our paper. Image tokenizers should be trained between images. We propose InterpTok, which mixes the codes of two training images at a known ratio and decodes the result. Each pair gives us an entire line of training inputs, even though no target image exists along that line. Training on these inputs creates two requirements. The decoded mixture must look like a natural image, and it must remain tied to its position between the two sources. Our first component, the *interpolation objective*, addresses both. An *interpolation discriminator* compares the decoded mixture with one source at a time in a fixed visual feature space and pushes it away from blurry overlays (Caron et al., 2021; Sauer et al., 2021). A *mixing ratio loss* prevents the decoder from choosing either source and keeps the result at the mixing ratio, in pixels and in the features of a frozen vision model (Siméoni et al., 2026). In the features, it connects the visual transition between two images to the semantic transition between them. Together, both terms turn the line between two codes into a training signal without requiring a target image.

Interpolation and reconstruction alone does not guarantee a latent space that a generator can learn easily. In our ablations, it improves the visual fidelity and sharpness of images between codes but harms both linear probe accuracy and generation quality. We therefore introduce a second component, *regularization losses*. They align the codes with the features of the same vision model, so that images with similar content receive similar codes (Yao et al., 2025). InterpTok combines the interpolation objective and the regularization losses with the original reconstruction objective of MAETok (Chen et al., 2025a).

Our contribution is a training principle for latent image tokenizers. We supervise the space between image codes while preserving the semantic structure at the codes themselves. On ImageNet, InterpTok reduces interpolated FID from 14.42 to 4.30 and generation FID of a matched generator

from 11.61 to 8.48. Its generator surpasses the final result of the MAETok generator after half of the training steps and reaches a generation FID of 8.11 with one eighth of the sampling steps. Most of this improvement comes from teaching the decoder to work between codes. When the codes and generator of MAETok remain unchanged, training only its decoder between codes recovers almost the entire improvement in generation, even though reconstruction becomes slightly worse. These results support a simple conclusion. A useful tokenizer needs codes that a generator can learn and a decoder that works in the space between them.

2 Related Work

Image tokenizers. The objective of image tokenizers is to compress images into a compact latent space and to decode the codes back to images. A generative model can then be trained in this compressed space, improving efficiency and performance (Rombach et al., 2022; Peebles & Xie, 2023). Early tokenizers were plain autoencoders, trained end to end for image compression and reconstruction (Hinton & Salakhutdinov, 2006). Variational autoencoders (VAEs) then add a penalty that keeps the code distribution close to a Gaussian (Kingma & Welling, 2014; Rombach et al., 2022). VQ-GAN improves overall visual quality by adding a discriminator that judges whether reconstructed images look natural (Esser et al., 2021). More recent work further targets the structure of the latent space instead of reconstruction alone, with 1D codes that drop the image grid (Yu et al., 2024) and with codes aligned to the features of pretrained vision models (Yao et al., 2025). MAETok (Chen et al., 2025a) showed that a diffusion model learns the codes more easily when they form fewer and more separated clusters, and reached this structure with a plain autoencoder trained by masked modeling, where auxiliary decoders predict features of the hidden image patches from the codes. Because masking degrades reconstruction, MAETok then freezes the encoder and fine-tunes only the pixel decoder for a few epochs, which recovers the lost detail.

Behaviour of generative denoising models. Generative denoising models do not simply recreate the image codes they saw during training. This can lead to generated codes not resembling what the decoder saw during training. These out-of-distribution codes are then decoded to images with blur or artifacts. Recent work therefore trains the decoder on codes other than the clean training codes, by transforming the code (Kouzelis et al., 2025), mixing it with noise (Yang et al., 2026b; Zheng et al., 2026) or post-training on the samples of a trained generator (Qiu et al., 2025). However, Aithal et al. (2024) showed that a diffusion model interpolates between nearby modes of its training data and produces samples that exist nowhere in it, which they call mode interpolation. Xu et al. (2026) carried this to tokenizer codes, where the optimal denoiser outputs a weighted average of the training codes (Karras et al., 2022). They show empirically that the quality of decoded mixtures of image codes (iFID) predicts the generation quality (gFID) far better, whereas reconstruction quality (rFID) is uncorrelated with it or even negatively correlated. Generated codes therefore resemble mixtures of image codes.

Supervising between images. Training an autoencoder on interpolation goes back to representation learning, where a realistic interpolation was taken as a sign that the autoencoder had captured the structure of its data instead of memorising it (Berthelot et al., 2019). ACAI trains the mixture to fool a critic that regresses the mixing coefficient (Berthelot et al., 2019), AMR and Oring et al. (2021) train it against a discriminator between real and mixed images (Beckham et al., 2019), and GAIA against an autoencoder that is trained to reconstruct real images but not mixtures (Sainburg et al., 2018). ACAI and AMR report smoother interpolations of small images and codes more useful for classification. Recently, CompTok (Zhao et al., 2026) argued that a generator learns a token space more easily when the images decoded along the line between the codes of two nearby images stay realistic, and measures this as mode connectivity. To raise it, CompTok trains on codes formed by swapping token subsets between two images under an adversarial realism term, but a swap composes real token values, whereas a generator produces values between two real ones (Xu et al., 2026), and CompTok does not train on the line it measures. Complementary to them, InterpTok carries the adversarial interpolation objective to a tokenizer and its diffusion generator, trains on the line that CompTok measures, and holds the decoded mixture at the mixing ratio with a mixing ratio loss.

3 Method

InterpTok is an image tokenizer trained to decode mixtures of image codes into sharp, natural images that combine both sources. It adds two components to the reconstruction objective of MAETok. The

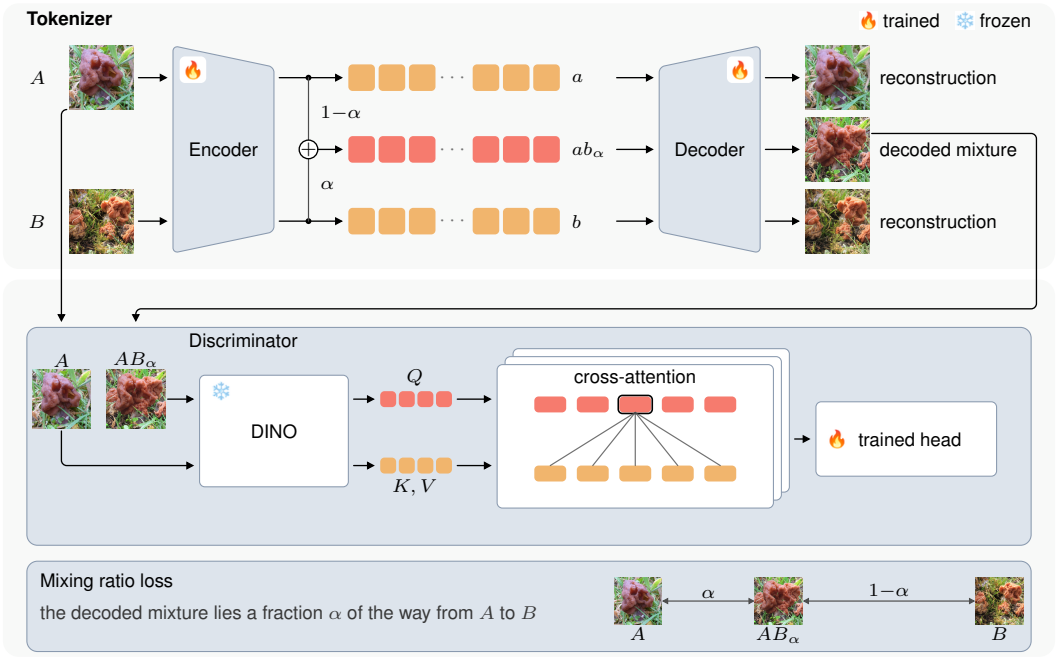


Figure 2: **Overview of InterpTok.** The encoder maps two images A and B to their codes a and b, and the decoder maps the codes and their mixture ab_α to two reconstructions and the decoded mixture AB_α. The interpolation discriminator judges the decoded mixture beside one of its sources, and the mixing ratio loss keeps it a fraction α of the way from A to B.

interpolation objective trains the decoder on mixtures of the codes of two images with an interpolation discriminator, which judges each decoded mixture beside one of its two source images, and a mixing ratio loss, which keeps the decoded mixture at the ratio at which the two codes were mixed. The regularization losses structure the latent space semantically. Fig. 2 gives an overview.

3.1 Background

Latent diffusion. Latent diffusion models generate images in the compact latent space of an image tokenizer instead of in pixel space, since training generative models on pixels is computationally too expensive (Rombach et al., 2022). The tokenizer consists of an encoder E and a decoder D. The encoder maps an image A to its code $a = E(A)$, a vector much smaller than the image itself. The decoder maps a code back to an image. Both are trained together, so that the decoded image $D(E(A))$ is close to the original image A. A diffusion generator is then trained separately on noisy image codes, with the task to denoise them. To generate an image, the generator starts from pure noise and iteratively refines it into a code, which D then decodes into the pixel image (Ho et al., 2020; Ma et al., 2024). Due to their training objective, diffusion generators tend to generate codes that lie between the codes of training images (Aithal et al., 2024; Xu et al., 2026). The decoder of the tokenizer, however, is trained only on the codes that the encoder produces for training images, leaving the codes between them unsupervised.

MAETok. We build on MAETok (Chen et al., 2025a), which shows that a diffusion model learns codes more easily when a Gaussian mixture model fits them with fewer modes. MAETok obtains such codes with a plain autoencoder, without the penalty of a variational autoencoder. Its code consists of 128 vectors, called tokens. MAETok is trained in two stages. In the first stage, a large part of every image is masked before it enters the encoder. Small auxiliary decoders then predict, from the code, the features that pretrained vision models extract from the masked parts, extending masked autoencoders (He et al., 2022) from pixels to features. Masking forces the encoder to infer the hidden parts of an image from the visible ones, which leads to such codes. However, a decoder trained on the codes of partly masked images reconstructs fewer details. To address this, the second stage freezes the encoder and fine-tunes only the decoder on the codes of complete images, which recovers the lost details.

We choose MAETok for three reasons. First, we want to show that our method is not limited to the

decoder, and therefore choose among tokenizers that train their own encoder. Among those in the benchmark of Xu et al. (2026), MAETok has the highest interpolation quality. We attribute this to its masked training, where the decoder is trained to reconstruct the complete image from the code of a partly masked image. Second, its codes are continuous. In a quantized tokenizer, every token is an index into a codebook, and there is no code between two indices. Third, MAETok is a 1D tokenizer (Yu et al., 2024), whose tokens are not tied to locations in the image, while every token of a 2D tokenizer belongs to one location. We want mixtures that interpolate between whole images, not between the patches at each location.

Reconstruction objective. MAETok trains with the standard tokenizer losses (Esser et al., 2021), a pixel loss, a perceptual loss (Zhang et al., 2018) and an adversarial loss. The pixel loss and the perceptual loss compare the decoded image with the original image, in pixels and in the features of a pretrained network. The adversarial loss trains the tokenizer against a discriminator that distinguishes real images from reconstructions. It pushes the decoded images toward the real images of the dataset. Unlike the other two losses, it needs no target image, since the discriminator judges each image on its own. We denote these losses together as the reconstruction objective $\mathcal{L}_{\text{rec}}$ and their discriminator as the reconstruction discriminator.

3.2 The Space Between Two Codes

We study the codes between the codes of two training images. Given two training images A and B, the sources, we obtain their codes $a = E(A)$ and $b = E(B)$. We form their mixture at ratio $\alpha \in [0, 1]$ and decode it,

$$ab_\alpha = (1 - \alpha)\, a + \alpha\, b, \qquad AB_\alpha = D(ab_\alpha). \tag{1}$$

We call ab_α the mixture and AB_α the decoded mixture. At $\alpha = 0$ and $\alpha = 1$ the mixture is the code of a source, and in between it moves along the line from a to b. Fig. 1 shows what MAETok's decoder produces on this line. Towards the middle, the decoded image superimposes both sources and is blurrier than either reconstruction. Xu et al. (2026) show that the quality of the images decoded between neighbouring training codes predicts generation quality better than reconstruction quality does. Recent methods add noise to the codes during decoder training (Yang et al., 2026b; Zheng et al., 2026), which makes the decoder more robust around each training code but does not target the space between codes.

A straightforward alternative is to supervise random codes, so that every code decodes to a realistic image. However, since real images have few intrinsic dimensions (Pope et al., 2021), their codes occupy only a small part of the latent space, and random codes mostly lie far outside the training distribution. Decoding a meaningful image from an arbitrary code is far more ambiguous than decoding one between two image codes. A mixture has no target image either, but it comes with two sources and a ratio, and InterpTok uses them in place of a target image.

Decodability and learnability. Two properties of a tokenizer matter for the quality of the output of a diffusion generator. We call the visual fidelity and sharpness of the images that the decoder produces from generated codes its decodability. We call how easily a conditioned generator can learn the codes of training images and tell them apart by their condition, such as their class label, its learnability. This is easier when the latent space is structured such that the class of an image can be derived from where its code lies, as MAETok finds. Decodability is thus mainly a property of the decoder, and learnability of the encoder. InterpTok targets decodability with the interpolation objective, Sec. 3.3, and learnability with the regularization losses, Sec. 3.4.

3.3 Interpolation Objective

The interpolation objective trains the decoder on decoded mixtures and targets decodability. It consists of an interpolation discriminator and a mixing ratio loss.

Interpolation discriminator. A mixture has no target image, and we therefore supervise the decoded mixture with a discriminator. Adversarial interpolation was introduced for representation learning with small autoencoders. ACAI (Berthelot et al., 2019) trains a critic to recover the mixing ratio from the decoded mixture and the autoencoder to hide it, and AMR (Beckham et al., 2019) trains a discriminator that tells real images from decoded mixtures, both on small images such as MNIST, SVHN and CIFAR-10. Their critics judge the mixture alone, so any realistic image passes, whether or not it relates to the two sources, and neither keeps the mixture at the mixing ratio. Our interpolation

discriminator judges the decoded mixture beside one of its sources and is trained to identify which of the two is the real image, and we use it to train the decoder of a tokenizer for latent diffusion on ImageNet at 256×256 resolution. For an image X and a second image Y, the features of X in the fixed feature space of a frozen pretrained network attend to those of Y through cross-attention, so that every part of X is compared with Y. A trained head maps the result to the score $s(X \mid Y)$, which is higher for images that look more like real images. Each decoded mixture is compared with both of its sources in turn. The discriminator is trained to score each source above the decoded mixture, and the tokenizer is trained to raise the score of the decoded mixture relative to that of its source, a relativistic objective (Jolicoeur-Martineau, 2019),

$$\mathcal{L}_{\text{disc}} = \frac{1}{2} \sum_{S \in \{A,B\}} \Big[s\big(S \mid AB_\alpha\big) - s\big(AB_\alpha \mid S\big) \Big]. \tag{2}$$

The decoder is thus trained to decode mixtures into images that look as real as their sources.

Mixing ratio loss. A decoder can fool the interpolation discriminator by reproducing one of the two sources. To overcome this, we introduce a mixing ratio loss, which requires the decoded mixture to be a fraction α of the way from A to B. It compares the distance of the decoded mixture to the reconstruction of each source,

$$\mathcal{L}_{\text{ratio}} = \left(\frac{d_A}{d_A + d_B} - \alpha \right)^2, \tag{3}$$

where $d_A = \|AB_\alpha - D(a)\|_1$ and $d_B = \|AB_\alpha - D(b)\|_1$ are the pixel distances to $D(a)$ and $D(b)$, the reconstructions at the two ends of the line. At $\alpha = \frac{1}{2}$, for instance, the decoded mixture has to be as far from one reconstruction as from the other. Pixel distance measures image content poorly, since two images of the same object can differ in almost every pixel. A second version, the feature mixing ratio loss, therefore compares the images in the features f of the frozen vision model. With θ the angle between two feature vectors, it requires $\theta\big(f(AB_\alpha), f(A)\big) = \alpha\,\theta\big(f(A), f(B)\big)$ and $\theta\big(f(AB_\alpha), f(B)\big) = (1-\alpha)\,\theta\big(f(A), f(B)\big)$. A crossfade, which lays the two sources over each other, satisfies the pixel version, and the interpolation discriminator penalizes it because it does not look like a real image. The discriminator penalizes visual crossfades and the mixing ratio loss penalizes trivial solutions such as copies. Together, they steer the decoder toward images that combine both sources.

3.4 Regularization Losses

The interpolation objective targets decodability but not learnability. We therefore add regularization losses that prior work uses to structure the latent space semantically. A frozen vision model, pretrained on large image collections, describes each image by a feature vector. Alignment losses make the codes carry the same semantic information. VA-VAE (Yao et al., 2025) aligns the code of a tokenizer with these features at every image position and requires the similarities between the positions of a code to match those between the features. REPA (Yu et al., 2025) requires the features to be predictable by a small network from the hidden states of a diffusion generator, and GigaTok (Xiong et al., 2025) from the activations inside the decoder of a tokenizer. InterpTok uses three alignment losses in this spirit, which require the similarities between the codes of different images to match those between their feature vectors, the feature vector to be predictable from the code, and the features of each image patch to be predictable from the activations inside the decoder. As a result, images with more similar visual and semantic features receive more similar codes, which these works find easier for a generator to learn. MAETok ties its codes to pretrained features as well, through auxiliary decoders that predict the features of the masked image parts, and InterpTok replaces these auxiliary decoders with the three alignment losses. The representation Fréchet loss (Yang et al., 2026a) was introduced for post-training generators, and to our knowledge it has not been used to train a tokenizer before. It compares the distribution of the decoded images with that of the training images in the features of frozen image encoders and minimizes the Fréchet distance between the two. The alignment losses act on single images and the representation Fréchet loss on all decoded images, mixtures included, and together they form $\mathcal{L}_{\text{reg}}$.

3.5 Full Objective

The full objective of InterpTok combines the reconstruction objective on single images, the interpolation objective on mixtures and the regularization losses,

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{disc}} + \mathcal{L}_{\text{ratio}} + \mathcal{L}_{\text{reg}}, \tag{4}$$

where App. E.2 gives the weight of every term.

4 EXPERIMENTS

We evaluate InterpTok on reconstruction, interpolation, generation and learnability, compare it with MAETok and with the tokenizers benchmarked by Xu et al. (2026), and ablate its components. Our experiments validate three claims. First, a modern tokenizer can be trained between images together with current regularization losses, and this improves interpolation quality, generation quality and learnability while keeping reconstruction quality. Second, the regularization losses alone do not suffice, and trained with them but without interpolation, the tokenizer reaches the highest gFID of all variants. Third, a conditioned generator needs both learnability, a property of the encoder, and decodability, a property of the decoder, and most of the generation gain comes from the decoder.

4.1 SETUP

Implementation details. We start from the released weights of MAETok and train its encoder and decoder together on ImageNet (Deng et al., 2009) at 256×256 resolution. The interpolation objective replaces MAETok's masking as a self-supervised objective. We evaluate on ImageNet and MS-COCO (Lin et al., 2014). We append six transformer blocks to the decoder, initialized as identity mappings. The feature mixing ratio loss and the alignment losses use DINOv3-B/16 (Siméoni et al., 2026) as the frozen vision model. We pair every training image with a similar image and mix the codes of every pair, since generated codes tend to lie between neighbouring training codes. For generation, we train a SiT-B (Ma et al., 2024) conditioned on class labels with the same setting on the codes of each tokenizer. We sample it without classifier-free guidance (Ho & Salimans, 2022), whose strength would have to be tuned for each tokenizer. More details are provided in App. E.

Metrics. For reconstruction, we compare each decoded image with its original, using the Peak Signal-to-Noise Ratio (PSNR) for pixel accuracy, the Structural Similarity Index (SSIM) for structural preservation and the Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al., 2018) for perceptual alignment. Decoded mixtures have no original to compare with. The Fréchet Inception Distance (FID) (Heusel et al., 2017), the standard metric for the quality of generated images, needs none, since it compares the distribution of a model's images with that of real images. Following Xu et al. (2026), we therefore measure decoded mixtures with the interpolated FID (iFID). iFID decodes the midpoint between each validation code and one of its nearest training codes and computes the FID of these images. It thus measures decodability. We also report the FID of reconstructions (rFID) and of generated images (gFID). We measure gFID at the usual 250 sampling steps and at fewer, since fewer steps make sampling cheaper. For learnability, we conduct linear probing as MAETok does, training a linear classifier to predict the class of an image from its code, and report its accuracy. We also report how often a code lies closest to the average code of its own class.

Table 1: **Reconstruction and interpolation on ImageNet.** InterpTok against nine of the thirteen tokenizers compared by Xu et al. (2026), and Table 29 lists the omitted rows. Among the nine are DM-VAE (Ye et al., 2026), REPA-E (Leng et al., 2025) and SoftVQ-VAE (Chen et al., 2025b). Latent dims counts the values of a code. Bold is the best value among trained encoders, and underlining marks an improvement of InterpTok over MAETok. †Quoted from that comparison or the tokenizer's own paper.

Tokenizer	Latent dims	rFID ↓	iFID ↓
Frozen DINOv2 encoder			
DM-VAE	8,192	0.683	8.29
RAE	196,608	0.49†	7.68†
Trained encoder			
SD-VAE	4,096	0.62†	59.91†
EQ-VAE	4,096	0.82†	47.51†
REPA-E	4,096	–	36.70†
SoftVQ-VAE	2,048	0.61†	26.91†
VA-VAE	8,192	**0.28**†	19.57†
l-DeTok	4,096	0.68†	17.51†
MAETok	4,096	0.582	14.42
InterpTok	4,096	0.597	**4.30**

4.2 MAIN RESULTS

Interpolation. In Table 1, InterpTok reduces the iFID of MAETok from 14.42 to 4.30, the lowest of all tokenizers benchmarked by Xu et al. (2026), and its midpoint FID on MS-COCO is six times lower than MAETok's, App. B.5. This demonstrates that decoding between codes can be learned, and App. A shows decoded paths for thirteen image pairs. The next best, RAE (Zheng et al., 2026), whose code consists of the features of the pretrained network DINOv2, reaches nearly twice our iFID. RAE and l-DeTok (Yang et al., 2026b) train their decoders on noisy codes, indicating that robustness to noise improves interpolation quality less than training between codes.

Table 3: **Ablations of InterpTok.** Interpolation is the interpolation discriminator with the pixel mixing ratio loss, and the vision model losses are the alignment losses and the feature mixing ratio loss. All other rows are fine-tuned from MAETok for 41,000 steps, except the row with interpolation and a frozen encoder, which uses only the interpolation discriminator and is fine-tuned for 13,000 steps. The two rows with a frozen encoder keep MAETok's codes and are evaluated with MAETok's generator. Bold is the best value in its column.

Tokenizer	Params (M)	iFID ↓	rFID ↓	probe ↑	gFID ↓
MAETok	174	14.42	0.582	73.0	11.61
with interpolation	174	5.44	0.556	63.5	18.90
frozen encoder	174	9.06	0.634	73.0	9.71
with regularization losses	217	12.86	0.588	77.0	21.15
InterpTok	217	**4.30**	0.597	**77.9**	**8.48**
without vision model losses	217	4.53	**0.544**	65.8	–
frozen encoder	217	4.46	0.637	73.0	8.56

Between the codes. Fig. 3a shows the FID of the images that MAETok and InterpTok decode from a code to its midpoint with one of its nearest training codes. At the code, both reach about the same FID. At the midpoint, MAETok's FID is 25 times its FID at the code, and InterpTok's only 7 times. Their decoders thus differ most between the codes, where reconstruction metrics are not computed, App. D. Fig. 3b shows that InterpTok reaches most of its gain at the midpoint within the first 13,000 fine-tuning steps.

Generation. Replacing MAETok with InterpTok reduces the gFID of the generator from 11.61 to 8.48, Table 2, in line with Xu et al. (2026), who find that iFID predicts gFID. With an eighth of the sampling steps, and thus an eighth of the sampling cost, InterpTok's generator reaches an even lower gFID, while the gFID of the generator on MAETok's codes rises with every reduction of the steps. InterpTok's generator also reaches the final gFID of MAETok's generator after only half of the training steps. We show metrics of generation and training evaluation in App. B.1, which also shows generated samples, Fig. 19.

Reconstruction. InterpTok keeps MAETok's reconstruction quality. Its rFID on ImageNet stays within three percent of MAETok's, and PSNR, SSIM and LPIPS improve on both ImageNet and MS-COCO, Table 11.

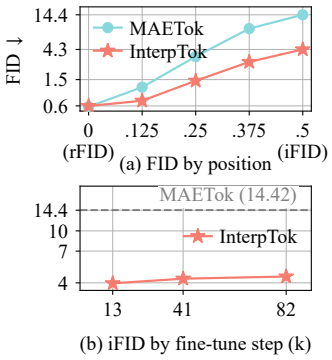


Figure 3: **FID between two codes.** (a) From a code to its midpoint with one of its nearest training codes, on a logarithmic axis. (b) The midpoint FID of InterpTok over its training steps.

Learnability. InterpTok's codes are more discriminative than MAETok's, although MAETok also ties its codes to pretrained features. A linear classifier predicts the class from InterpTok's codes more accurately, Table 3, and 73 percent of InterpTok's codes lie closest to the average code of their own class, against 55 percent of MAETok's, App. B.9. The class of an image can thus be derived from its code more reliably, and a generator conditioned on class labels can learn more easily which codes belong to which class, App. B.3 shows the codes of both encoders.

4.3 Ablations

Table 3 compares variants of MAETok and of InterpTok, all trained on the same data. Two variants add interpolation or the regularization losses to MAETok and train encoder and decoder. A third adds only the interpolation discriminator and trains only the decoder, with MAETok's encoder frozen. One variant of InterpTok removes every loss that uses the frozen vision model, the three alignment losses and the feature mixing ratio loss, and another trains only the decoder, again with MAETok's encoder frozen.

Table 2: **Generation against sampling steps.** Each generator is a class-conditional SiT-B trained for 400,000 steps and sampled without guidance with the stochastic sampler of SiT (Ma et al., 2024). The second row decodes the first row's generated codes, from the same noise, with the decoder that Table 3 trains on MAETok's frozen encoder. $n = 50{,}000$ per cell. Bold is the best value in its column.

gFID ↓	250	64	32	16
MAETok	11.61	12.33	13.88	20.56
with our decoder	8.56	8.99	9.92	15.33
InterpTok	**8.48**	**8.09**	**8.11**	**12.48**

Interpolation and regularization. Trained with interpolation but without the regularization losses, the tokenizer improves interpolation quality nearly as much as InterpTok, but a linear classifier predicts the class from its codes less accurately than from MAETok's. Trained with the regularization losses but without interpolation, it raises the probe accuracy but barely improves interpolation quality, and its gFID is the highest of all variants. With only interpolation or only the regularization losses, the tokenizer reaches a higher gFID than MAETok, while InterpTok, which combines both, reaches a lower one, indicating that a generator needs both learnability and decodability. Without the losses that use the vision model, InterpTok keeps most of its interpolation gain, but its probe accuracy falls below MAETok's, indicating that most of the learnability comes from these losses.

Decoder alone. The variant of InterpTok that trains only the decoder keeps MAETok's codes, and MAETok's generator can therefore be used with its decoder without retraining. With this decoder, the gFID of MAETok's generator comes close to that of InterpTok, Table 2. This shows that most of the generation gain comes from the decoder, and that an existing generator can be improved by replacing only its decoder, which also improves the larger SiT-XL released with MAETok, App. B.2. The variant that adds only the interpolation discriminator to MAETok's losses already gives more than half of this gain, while a decoder trained with MAETok's losses alone gives about a quarter, App. B.9.

Latent space. With the regularization losses but without interpolation, a code's nearest neighbour belongs to its own class twice as often as with MAETok's encoder, but the codes vary along three times as many directions, Table 21. With InterpTok's encoder, the nearest neighbour belongs to its own class about as often, while the codes vary along a third fewer directions. A generator models every direction along which the codes vary, and a latent space with fewer effective dimensions is easier for a diffusion model to learn (Zhong et al., 2026; Yao et al., 2025).

Fewer sampling steps. We hypothesize that the latent space also explains why InterpTok's generator reaches a lower gFID with 64 or 32 sampling steps than with 250. With fewer steps, a generated code lies closer to the average code of its class. With InterpTok's encoder, the codes near this average belong to that class more than twice as often as with MAETok's, and the generated images therefore match their class more often. InterpTok's decoder, trained between codes, also decodes such averages with a lower FID than MAETok's decoder, App. B.9. Both lower the gFID.

5 Conclusion

This paper focuses on the space between the codes of training images, where diffusion generators tend to generate codes and where the quality of the decoded images predicts generation quality (Xu et al., 2026). Adding noise to the codes during decoder training makes the decoder more robust but does not target this space. To address this, we presented InterpTok, which trains the tokenizer on mixtures of the codes of two training images. An interpolation discriminator judges each decoded mixture beside one of its sources, a mixing ratio loss keeps it at the mixing ratio between them, and regularization losses structure the latent space semantically. On MAETok, InterpTok reduces the iFID more than threefold, to the lowest among the tokenizers benchmarked by Xu et al. (2026), and the gFID by more than a quarter, while keeping reconstruction quality. Its generator reaches an even lower gFID with an eighth of the sampling steps, and the final gFID of MAETok's generator after half of the training steps. Most of the generation gain comes from the decoder, and replacing only the decoder improves an existing generator without retraining it. Trained with interpolation but without regularization, or the reverse, the tokenizer reaches a higher gFID than MAETok, and without interpolation the highest of all variants. A conditioned generator thus needs both learnability, a property of the encoder, and decodability, a property of the decoder. Training between images opens several directions for future work. For example, a much smaller code trained with the interpolation objective leads to more semantic codes and more semantic interpolations, App. C. Decoders are much faster than generative models, so future work could investigate this self-supervised training objective for real-time animation, video generation or granular editing.

Our findings show that the space between image codes can and should be explicitly supervised rather than ignored.

Reproducibility statement

App. E and Table 25 give every hyperparameter, loss and schedule. Interpolation and reconstruction are measured with the evaluation implementation released by Xu et al. (2026) at its defaults, and

Table 27 lists every dataset and pairing rule. We will release the implementation of training and evaluation, the configuration of every run and the tokenizer weights of the 41,000-step and 82,000-step checkpoints of App. B.4.

ETHICS STATEMENT

This work improves image tokenizers for generation and editing. We train and evaluate on ImageNet and MS-COCO under their research terms and collect no human-subject data. The images at the ends of the paths in App. C and at the corners of its squares were generated by a text-to-image model.

AI USE STATEMENT

Large language model assistants supported this work in finding related work, in the research, in writing code and in drafting and editing the text. We reviewed everything they produced and take responsibility for the content.

REFERENCES

Sumukh K Aithal, Pratyush Maini, Zachary C. Lipton, and J. Zico Kolter. Understanding hallucinations in diffusion models through mode interpolation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2406.09358.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations*, 2018. arXiv:1710.11379.

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. FlexTok: Resampling images into 1D token sequences of flexible length. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 2241–2292, 2025. arXiv:2502.13967.

Christopher Beckham, Sina Honari, Vikas Verma, Alex Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Christopher Pal. On adversarial mixup resynthesis. In *Advances in Neural Information Processing Systems*, volume 32, 2019. arXiv:1903.02709.

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019. arXiv:1807.07543.

Swayam Bhanded. Speedrunning ImageNet diffusion. *arXiv preprint arXiv:2512.12386*, 2025.

Giulio Biroli, Tony Bonnaire, Valentin de Bortoli, and Marc Mézard. Dynamical regimes of diffusion models. *Nature Communications*, 15:9957, 2024. arXiv:2402.18491.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9650–9660, 2021. arXiv:2104.14294.

Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 8145–8171, 2025a. arXiv:2502.03444.

Hao Chen, Ze Wang, Xiang Li, Ximeng Sun, Fangyi Chen, Jiang Liu, Jindong Wang, Bhiksha Raj, Zicheng Liu, and Emad Barsoum. SoftVQ-VAE: Efficient 1-dimensional continuous tokenizer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28358–28370, 2025b. arXiv:2412.10958.

Zhengdao Chen. On the interpolation effect of score smoothing in diffusion models. In *International Conference on Learning Representations*, 2026. arXiv:2502.19499.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, 2021. arXiv:2105.05233.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021. arXiv:2012.09841.

Weichen Fan, Haiwen Diao, Quan Wang, Dahua Lin, and Ziwei Liu. The prism hypothesis: Harmonizing semantic and pixel representations via unified autoencoding. In *European Conference on Computer Vision*, 2026. arXiv:2512.19693.

Jing Gu, Morteza Mardani, Wonjun Lee, Dongmian Zou, and Gilad Lerman. Understanding latent diffusability via Fisher geometry. *arXiv preprint arXiv:2604.02751*, 2026.

Jiayi Guo, Xingqian Xu, Yifan Pu, Zanlin Ni, Chaofei Wang, Manushree Vasu, Shiji Song, Gao Huang, and Humphrey Shi. Smooth diffusion: Crafting smooth latent spaces in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. arXiv:2312.04410.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. arXiv:2111.06377.

Qiyuan He, Jinghao Wang, Ziwei Liu, and Angela Yao. AID: Attention interpolation of text-to-image diffusion. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2403.17924.

Ye He, Yitong Qiu, and Molei Tao. Diffusion model's generalization can be characterized by inductive biases toward a data-dependent ridge manifold. *arXiv preprint arXiv:2602.06021*, 2026.

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. In *Advances in Neural Information Processing Systems*, volume 30, 2017. arXiv:1706.08500.

Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020. arXiv:2006.11239.

Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking FID: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9307–9315, 2024. arXiv:2401.09603.

Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard GAN. In *International Conference on Learning Representations*, 2019. arXiv:1807.00734.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019. arXiv:1812.04948.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. arXiv:2206.00364.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

Theodoros Kouzelis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. EQ-VAE: Equivariance regularized latent space for improved generative image modeling. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 31648–31666, 2025.

Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. REPA-E: Unlocking VAE for end-to-end tuning of latent diffusion transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. arXiv:2504.10483.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

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 *European Conference on Computer Vision*, 2024. arXiv:2401.08740.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations*, 2018. arXiv:1802.05637.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. arXiv:1802.05957.

Alon Oring, Zohar Yakhini, and Yacov Hel-Or. Autoencoder image interpolation by shaping the latent space. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *PMLR*, pp. 8281–8290, 2021. arXiv:2008.01487.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. arXiv:2212.09748.

Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021.

Kai Qiu, Xiang Li, Hao Chen, Jason Kuen, Xiaohao Xu, Jiuxiang Gu, Yinyi Luo, Bhiksha Raj, Zhe Lin, and Marios Savvides. Image tokenizer needs post-training. *arXiv preprint arXiv:2509.12474*, 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 *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 8748–8763, 2021. arXiv:2103.00020.

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*, 2022.

Tim Sainburg, Marvin Thielk, Brad Theilman, Benjamin Migliori, and Timothy Gentner. Generative adversarial interpolative autoencoding: Adversarial training on latent space interpolations encourage convex latent distributions. *arXiv preprint arXiv:1807.06650*, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems*, volume 29, 2016. arXiv:1606.03498.

Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected GANs converge faster. In *Advances in Neural Information Processing Systems*, volume 34, 2021. arXiv:2111.01007.

Antonio Sclocchi, Alessandro Favero, and Matthieu Wyart. A phase transition in diffusion models reveals the hierarchical nature of data. *Proceedings of the National Academy of Sciences*, 122(1): e2408799121, 2025. arXiv:2402.16991.

Mohamad Shahbazi, Martin Danelljan, Danda Pani Paudel, and Luc Van Gool. Collapse by conditioning: Training class-conditional GANs with limited data. In *International Conference on Learning Representations*, 2022. arXiv:2201.06578.

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

Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *PMLR*, pp. 55876–55905, 2025. arXiv:2502.14831.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. arXiv:2011.13456.

Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularizing generative adversarial networks under limited data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. arXiv:2104.03310.

Clinton J. Wang and Polina Golland. Interpolating between images with diffusion models. In *ICML Workshop on Challenges of Deploying Generative AI*, 2023. arXiv:2307.12560.

Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

Tianwei Xiong, Jun Hao Liew, Zilong Huang, Jiashi Feng, and Xihui Liu. GigaTok: Scaling visual tokenizers to 3 billion parameters for autoregressive image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Tongda Xu, Mingwei He, Shady Abu-Hussein, José Miguel Hernández-Lobato, Chunhang Zheng, Kai Zhao, Chao Zhou, Ya-Qin Zhang, and Yan Wang. Making reconstruction FID predictive of diffusion generation FID. *arXiv preprint arXiv:2603.05630*, 2026.

Yilun Xu, Mingyang Deng, Xiang Cheng, Yonglong Tian, Ziming Liu, and Tommi Jaakkola. Restart sampling for improving generative processes. In *Advances in Neural Information Processing Systems*, volume 36, 2023. arXiv:2306.14878.

Jiawei Yang, Zhengyang Geng, Xuan Ju, Yonglong Tian, and Yue Wang. Representation Fréchet loss for visual generation. *arXiv preprint arXiv:2604.28190*, 2026a.

Jiawei Yang, Tianhong Li, Lijie Fan, Yonglong Tian, and Yue Wang. Latent denoising makes good tokenizers. In *International Conference on Learning Representations*, 2026b. arXiv:2507.15856.

Zhaoyuan Yang, Zhengyang Yu, Zhiwei Xu, Jaskirat Singh, Jing Zhang, Dylan Campbell, Peter Tu, and Richard Hartley. IMPUS: Image morphing with perceptually-uniform sampling using diffusion models. In *International Conference on Learning Representations*, 2024.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. arXiv:2501.01423.

Sen Ye, Jianning Pei, Mengde Xu, Shuyang Gu, Chunyu Wang, Liwei Wang, and Han Hu. Distribution matching variational autoencoder. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. arXiv:2512.07778.

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2406.07550.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025. arXiv:2410.06940.

Zhengrong Yue, Taihang Hu, Mengting Chen, Haiyu Zhang, Zihao Pan, Tao Liu, Zikang Wang, Jinsong Lan, Xiaoyong Zhu, Bo Zheng, and Yali Wang. What matters for diffusion-friendly latent manifold? Prior-aligned autoencoders for latent diffusion. *arXiv preprint arXiv:2605.07915*, 2026.

Z-Image Team, Huanqia Cai, Sihan Cao, et al. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *International Conference on Learning Representations*, 2020. arXiv:1910.12027.

Kaiwen Zhang, Yifan Zhou, Xudong Xu, Xingang Pan, and Bo Dai. DiffMorpher: Unleashing the capability of diffusion models for image morphing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. arXiv:2312.07409.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. arXiv:1801.03924.

Bingchen Zhao, Qiushan Guo, Ye Wang, Yixuan Huang, Zhonghua Zhai, and Yu Tian. Composable visual tokenizers with generator-free diagnostics of learnability. *arXiv preprint arXiv:2602.03339*, 2026.

Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient GAN training. In *Advances in Neural Information Processing Systems*, volume 33, 2020. arXiv:2006.10738.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, 2026. arXiv:2510.11690.

Pengfei Zheng, Yonggang Zhang, Zhen Fang, Tongliang Liu, Defu Lian, and Bo Han. NoiseDiffusion: Correcting noise for image interpolation with diffusion models beyond spherical linear interpolation. In *International Conference on Learning Representations*, 2024.

Tianxiong Zhong, Xingye Tian, Xuebo Wang, Xin Tao, and Pengfei Wan. Diffusing in the right space: A systematic study of latent diffusability. *arXiv preprint arXiv:2606.03578*, 2026.

OVERVIEW

The appendix first shows further results and then the details behind them. App. A shows decoded paths between thirteen image pairs. App. B reports generated samples and generation quality over the generator's training, the latent space of each encoder, reconstruction on ImageNet and MS-COCO, mixtures of several codes, and why generation quality changes with the number of sampling steps. App. C describes an earlier experiment with a much smaller code size, whose decoded mixtures are more semantic compositions of the scenes. App. D states the formal observations behind the method, and App. E gives the implementation details, the training objectives and the evaluation protocol.



Figure 4: **Selected interpolation, agaric.**



Figure 5: **Selected interpolation, hip.**

A Additional Visualizations

Figures 4–16 show thirteen selected pairs at all nine positions, one pair per figure. In Figures 4–16, the rows are, from top to bottom, the released MAETok, SD-VAE, and InterpTok after 13,000, 41,000 and 82,000 fine-tuning steps. The sources are ImageNet validation images, which neither MAETok nor InterpTok trained on.

Mixtures of three images. Although InterpTok trains on mixtures of two codes, it also decodes averages of several codes with a lower FID than MAETok, App. B.6. In Fig. 17, MAETok's decoder decodes mixtures of three mountain images into overlays of the sources, while InterpTok's decoder produces single mountain scenes.

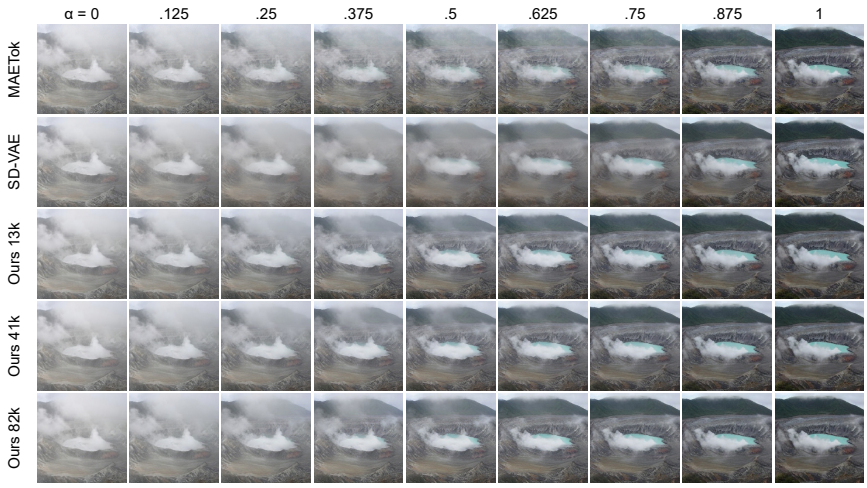


Figure 6: **Selected interpolation, volcano.**

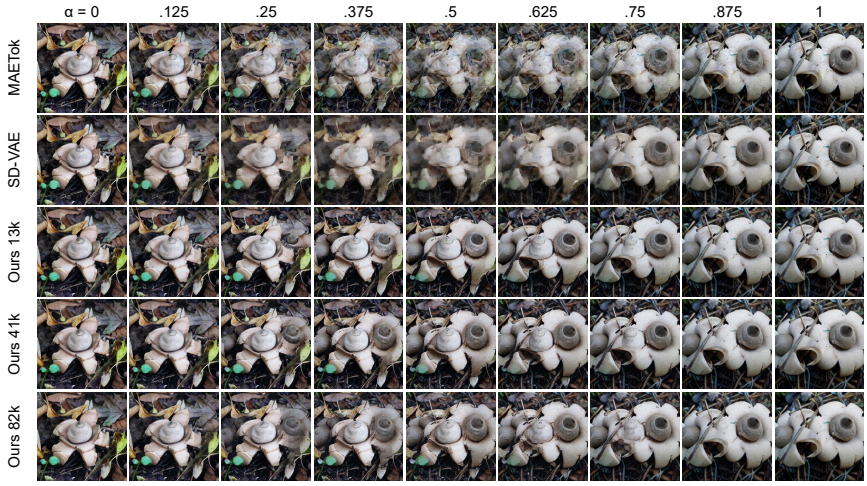


Figure 7: **Selected interpolation, earthstar, first pair.**

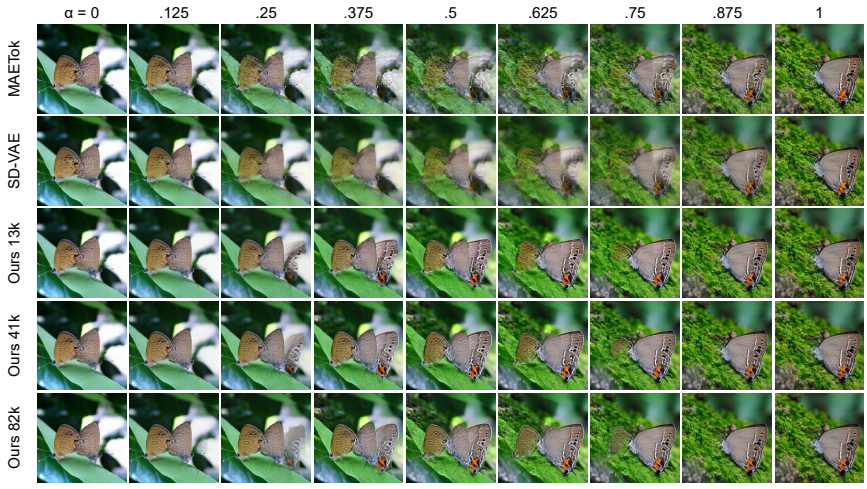


Figure 8: **Selected interpolation, lycaenid.**

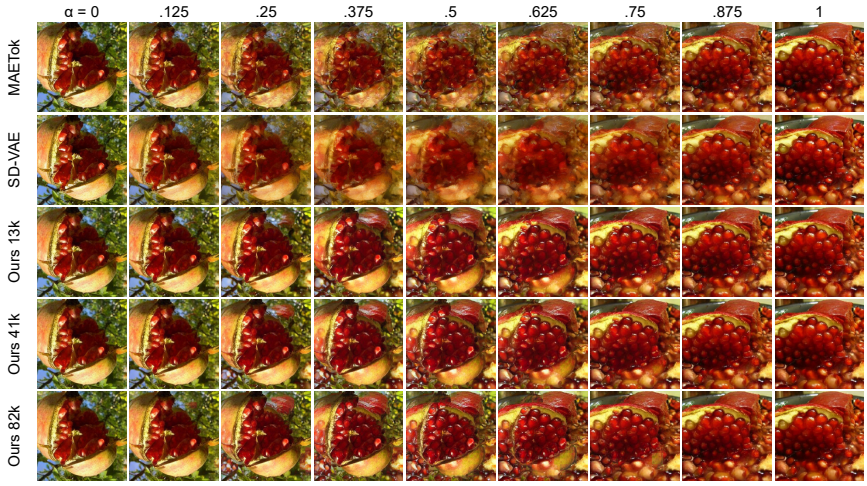


Figure 9: **Selected interpolation, pomegranate.**



Figure 10: **Selected interpolation, earthstar, second pair.**

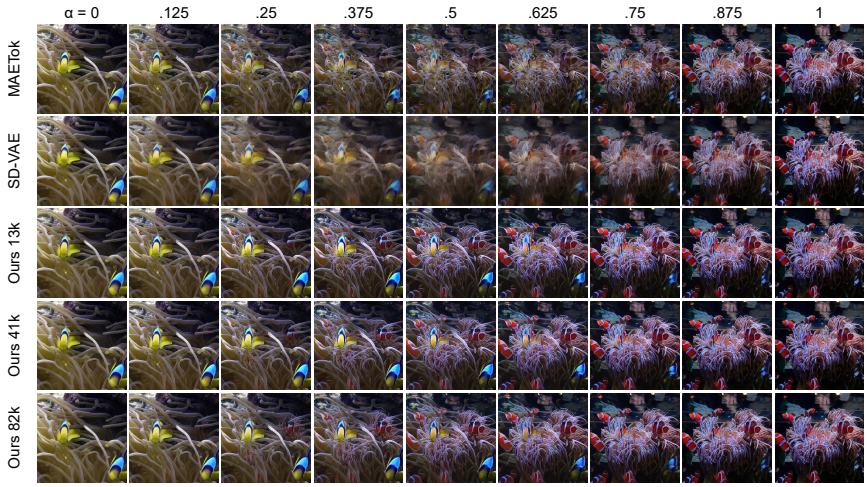


Figure 11: **Selected interpolation, anemone fish.**

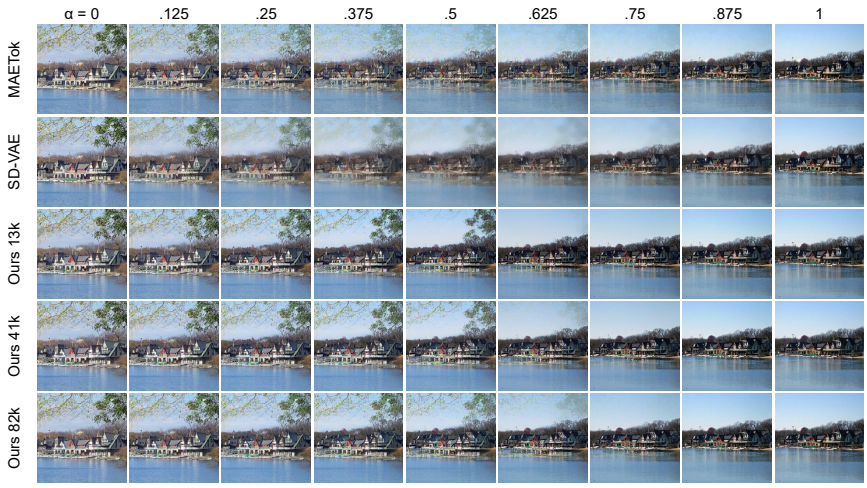


Figure 12: **Selected interpolation, boathouse.**

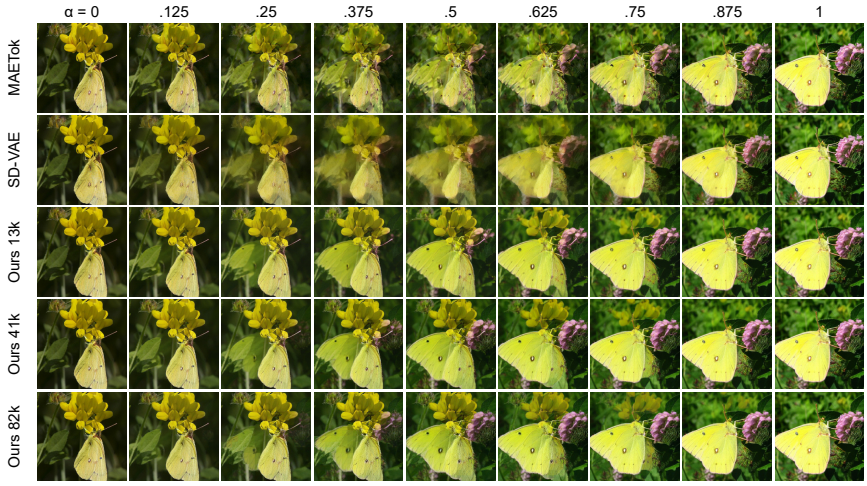


Figure 13: **Selected interpolation, sulphur butterfly.**

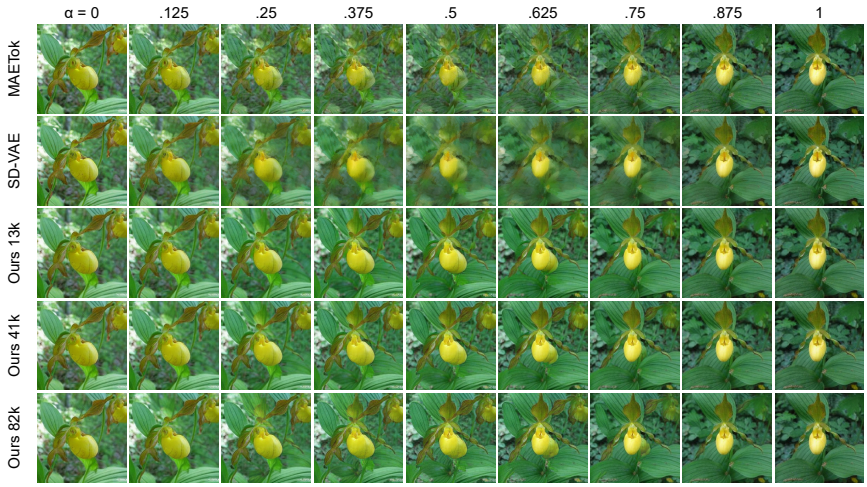


Figure 14: **Selected interpolation, yellow lady's slipper.**

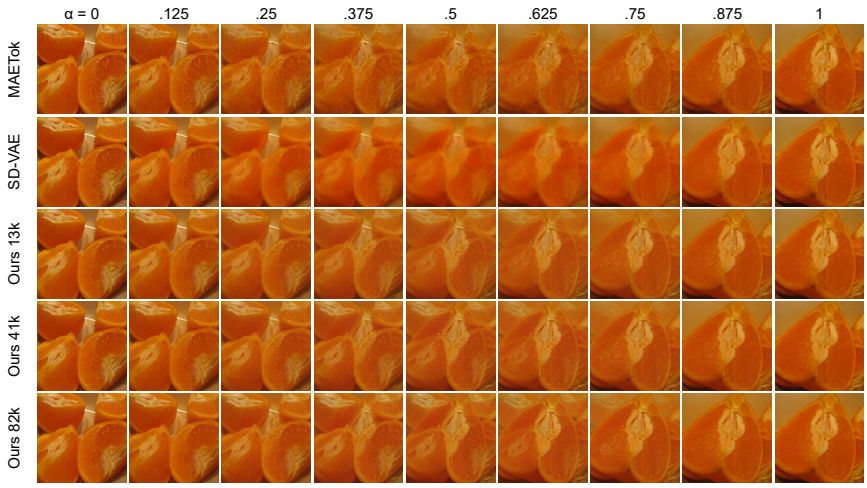


Figure 15: **Selected interpolation, orange.**

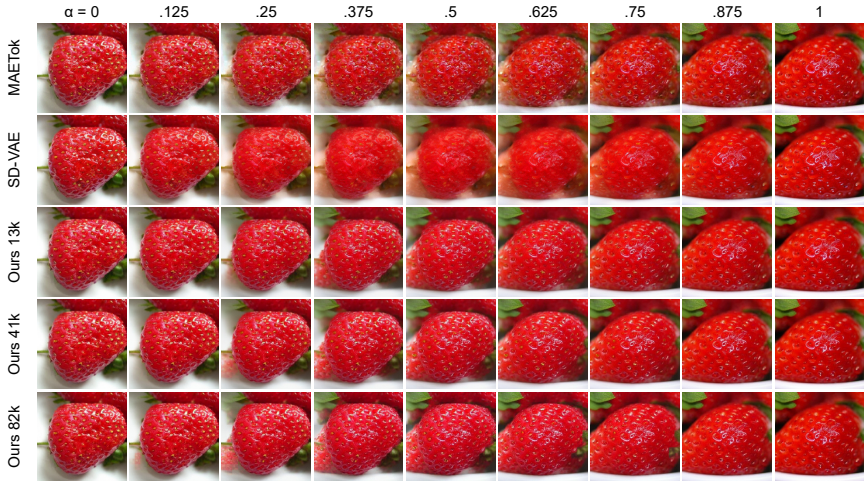


Figure 16: **Selected interpolation, strawberry.**

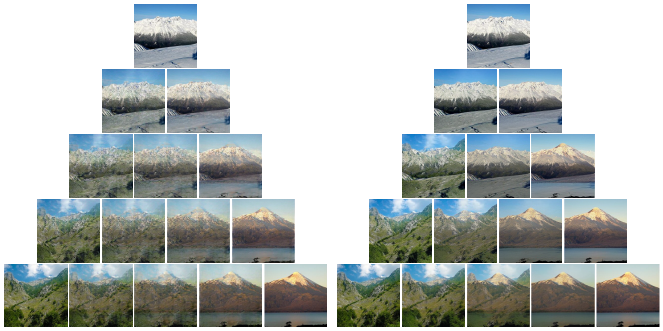


Figure 17: **Selected three-image interpolation.** MAETok left and InterpTok right, with the source images at the corners, mixtures of two neighbouring corners along the edges in steps of a quarter and mixtures of all three inside.

B Additional Results

B.1 Generation

Every SiT-B/1D result in this paper comes from generators we trained under one recipe, and our MAETok generator is the control against which InterpTok is compared. This control reaches a gFID of 11.61, lower than the 13.73 that Xu et al. (2026) release for a MAETok generator of this size trained for 400,000 steps at global batch 128. Table 4 reports our SiT-B/1D generators (Ma et al., 2024), trained in the released pipeline of Xu et al. (2026), beside published configurations. Fig. 18 follows them across the generator's training, Table 5 scores the same samples in four feature spaces, and Fig. 19 shows selected samples. InterpTok leads to a lower gFID than MAETok at every generator budget we measured, with the gap widest early and settling near a quarter, Fig. 18. Both fine-tuned checkpoints lead to a lower Fréchet distance than MAETok, the 41,000-step one in all four feature spaces and the 82,000-step one in three of four. SD-VAE reaches a higher PSNR and SSIM and a lower LPIPS than MAETok, but a higher gFID.

Table 4: **Generation across generator training budgets, beside the published configurations.** On the left, one SiT-B/1D per row on the named tokenizer's codes, sampled at 250 steps without guidance. On the right, values as published, sampled without guidance, whose generator size, patch size and training budget differ across rows. $n = 50{,}000$ per cell.

Tokenizer	generator steps	gFID ↓
MAETok	100k	24.07
InterpTok	100k	16.26
MAETok	200k	15.88
InterpTok	200k	11.27
MAETok	300k	13.04
InterpTok	300k	9.54
MAETok	400k	11.61
InterpTok	400k	8.48

Tokenizer	Published generator	Steps	gFID
SD-VAE	SiT-B/2 (Ma et al., 2024; Bhanded, 2025)	400k	33.0
MAETok	SiT-L (Chen et al., 2025a)	400k	5.69
MAETok	SiT-XL (Chen et al., 2025a)	4M	2.31

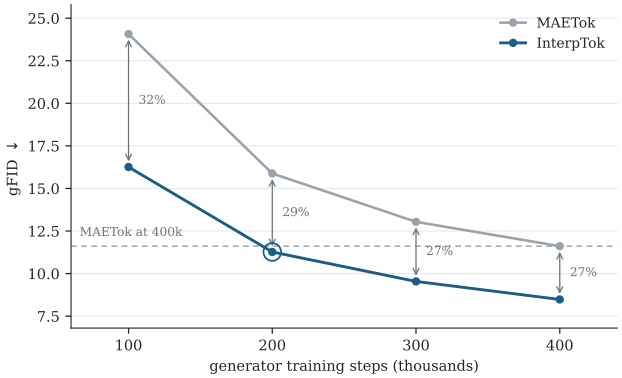


Figure 18: **Generation across the generator's training.** The percentages are the gap at each budget and the dashed rule is MAETok's value at 400,000 steps.

Table 5: **Generated samples in four feature spaces.** Each row is a SiT-B/1D at 400,000 steps on the named tokenizer's codes, against the 50,000 ImageNet validation images and, for ADM FID, against the ADM reference statistics. Values are comparable within a column only.

Tokenizer	SigLIP2 ↓	DINOv2 ↓	MAE ↓	Inception pool ↓	ADM FID ↓
MAETok	63.72	344.8	0.825	13.20	11.61
InterpTok (41k)	59.01	309.6	0.775	10.27	8.48
InterpTok (82k)	60.73	323.1	0.851	11.69	9.57

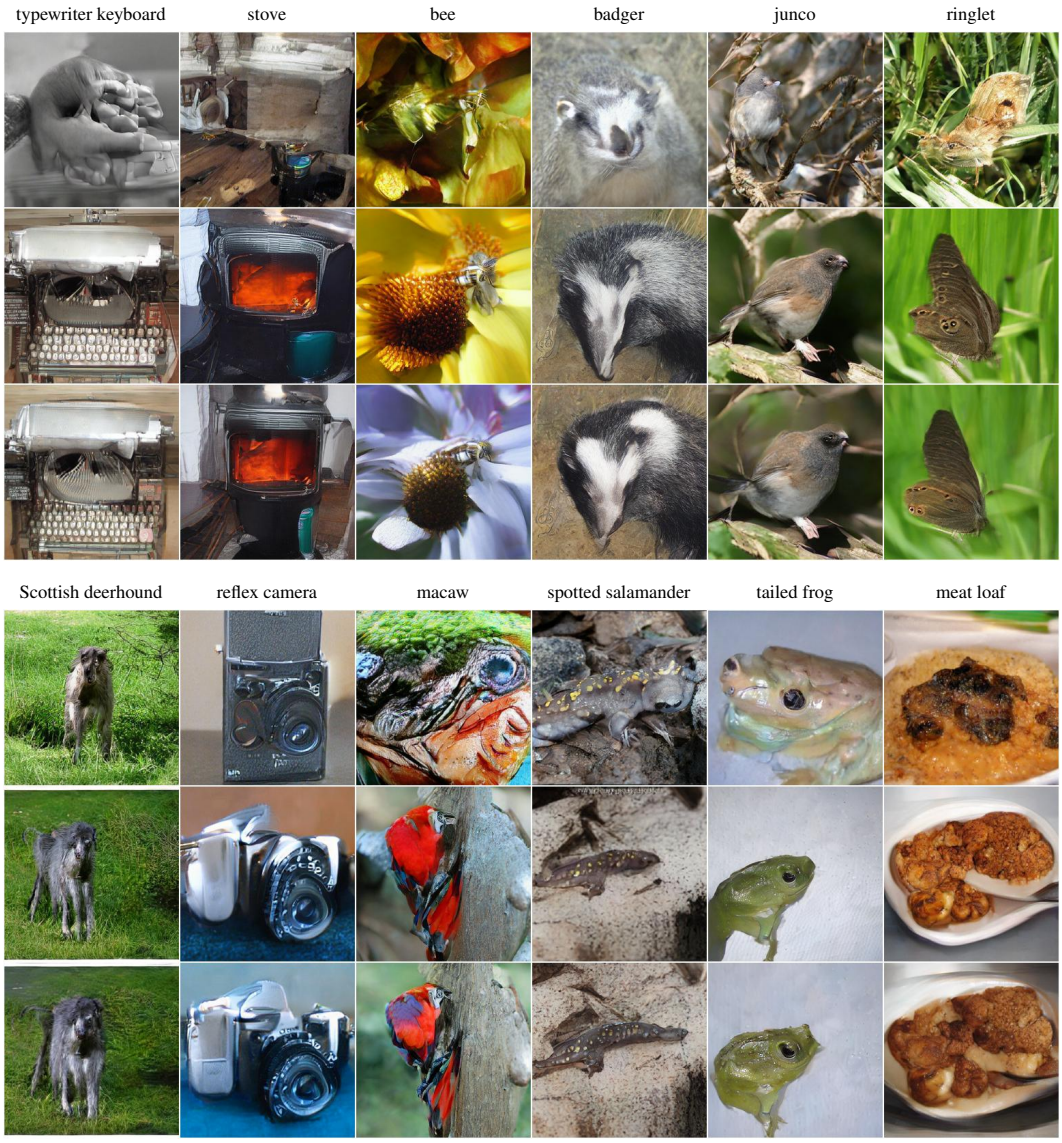


Figure 19: **Selected samples without guidance.** In each block of three rows, MAETok above, InterpTok (41k) in the middle and InterpTok (82k) below. Columns share a class and an initial noise.

B.2 Decoding a fixed generator's samples

Table 6 decodes the same generated codes with MAETok's decoder and with ours. For our SiT-B on MAETok's codes, the gain holds in all four feature spaces. For the released SiT-XL, which we did not train, we score the swap against our own draw of that generator, which measures 2.46 where its authors publish 2.31. In Table 2, the generator, the starting noise and the sampler are the same within every column, and the advantage of our decoder grows with every reduction of the sampling steps.

Table 6: **One generator, two decoders.** $n = 50{,}000$ samples per block, decoded by MAETok's decoder and by our decoder, comparable within a column only.

our SiT-B on MAETok's codes

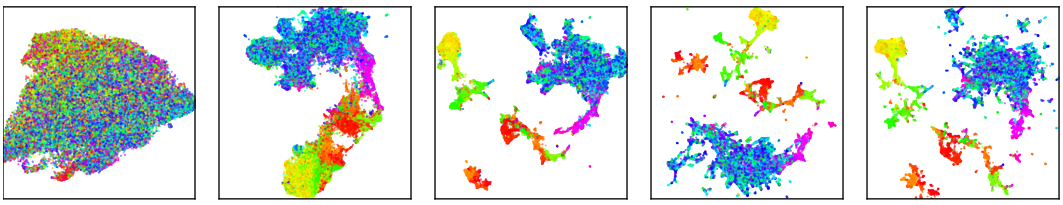
Decoder	FID ↓	SigLIP2 ↓	DINOv2 ↓	MAE ↓	Inception ↓
MAETok	11.61	63.72	344.8	0.825	13.20
+ our decoder	8.56	55.97	274.3	0.669	9.82

the released MAETok SiT-XL (Chen et al., 2025a)

Decoder	FID ↓
MAETok	2.46
+ our decoder	2.26

B.3 IMAGE REPRESENTATIONS

Table 21 gives the linear-probe accuracy on flattened codes under the MAE procedure of Table 27, and the probe column of Table 8 follows it across the fine-tune. Under the same procedure, a linear probe predicts the class from SD-VAE's codes far less accurately than from MAETok's. RAE reports a higher accuracy for its frozen foundation encoder under a different procedure, frozen DINOv2-B linear probing. Fig. 20 projects the validation codes of each tokenizer with the released implementation of UMAP (McInnes et al., 2018) at its default settings, with one window shape per panel so that apparent density is comparable. Its three right-hand panels show three checkpoints of one fine-tuning run.



SD-VAE | MAETok | Ours 13k | Ours 41k | Ours 82k

Figure 20: **UMAP projections of image codes.** $n = 50{,}000$ ImageNet validation codes per panel, coloured by class.

B.4 EVALUATION DURING TRAINING

Table 8 follows four checkpoints of one fine-tuning run, whose two pair sets differ in pairing, geometry and reference images. The random-pair midpoint FID falls more than sixfold from the released model to 41,000 steps and changes little afterwards. The reconstruction FID keeps falling after 13,000 steps, and the linear probe keeps improving. Both the 41,000-step and the 82,000-step checkpoints lead to a lower gFID than MAETok. We report 41,000 steps, since the generator on these codes reaches the lowest gFID and every variant of Table 3 except the one with only the interpolation discriminator is matched there. The 13,000-step checkpoint has the lowest iFID, but a linear classifier predicts the class from its codes less accurately than from MAETok's. A generator on its codes reaches a higher gFID than one on the 41,000-step codes. Generation thus depends on learnability as well as on the decodability that iFID measures, in line with Zhong et al. (2026).

Table 7: **Generation from tokenizer codes.** Matched SiT-B/1D generators at 250 sampling steps without guidance, $n = 50{,}000$.

Tokenizer	gFID ↓
MAETok	11.61
InterpTok (13k)	8.94
InterpTok (41k)	**8.48**
InterpTok (82k)	9.57

Table 8: **Checkpoints of the fine-tune**, $n = 50{,}000$ per FID.

Cross-class pairs	MAETok	13k	41k	48k	82k
midpoint FID ↓	87.34	16.06	13.88	13.86	14.92
reconstruction FID ↓	0.575	0.581	0.574	0.561	0.553

Nearest-neighbour pairs	MAETok	13k	41k	82k
iFID ↓	14.42	3.97	4.30	4.47
rFID ↓	0.582	0.601	0.597	0.574

Codes	MAETok	41k	82k
probe ↑	73.0	77.9	78.4
gFID ↓	11.61	8.48	9.57

B.5 RECONSTRUCTION ON IMAGENET AND MS-COCO

The main results pair each code with one of its nearest training codes and decode their midpoint. Here, we vary the pairing, the position on the line, the number of mixed codes and the dataset. InterpTok has higher interpolation quality than MAETok under every pairing, number of codes and dataset we measure, with the largest advantage near the middle of the line.

Tables 11, 9 and 10 report reconstruction on ImageNet and MS-COCO and the decoded MS-COCO midpoints in three feature spaces. On MS-COCO, where neither tokenizer is trained, InterpTok's midpoint FID is six times lower than MAETok's. The variant trained without the regularization losses has an even lower midpoint FID. Among these tokenizers, SD-VAE has the lowest reconstruction FID and the highest midpoint FID.

Table 9: **MS-COCO reconstruction and decoded midpoints.** FID against the 50,000 train2017 sources, $n = 50{,}000$ per cell.

Tokenizer	Reconstruction ↓	Midpoint ↓
SD-VAE	**0.801**	135.89
MAETok	0.863	85.21
+ interpolation	0.865	**9.23**
InterpTok	1.013	13.61

In the features of DINOv2, which MAETok's auxiliary decoders regress and none of our losses uses, InterpTok's midpoint Fréchet distance is still three times lower. The 41,000-step checkpoint reaches a lower Fréchet distance than the 82,000-step one in the SigLIP2 and MAE features, which the representation Fréchet loss uses, and both reach about the same in DINOv2.

Table 10: **Decoded MS-COCO midpoints in three feature spaces**, $n = 50{,}000$ midpoints of random pairs of different train2017 images, against the COCO sources. Values are comparable within a column only.

Tokenizer	SigLIP2 ↓	MAE ↓	DINOv2 ↓
MAETok	85.8	2.03	953
InterpTok (41k)	34.6	0.52	315
InterpTok (82k)	36.4	0.70	311

Table 11: **Reconstruction on ImageNet and MS-COCO.** The ImageNet validation set and MS-COCO train2017, $n = 50{,}000$ per cell. †Quoted from Chen et al. (2025a).

			ImageNet				MS-COCO			
Tokenizer	Params (M)	Tokens	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓	rFID ↓	PSNR ↑	SSIM ↑	LPIPS ↓
SD-VAE	84†	1024	0.62†	**25.00**	**0.805**	**0.0630**	**0.801**	**24.76**	**0.817**	**0.0609**
MAETok	174	128	**0.582**	23.55	0.760	0.0917	0.863	23.25	0.770	0.0913
InterpTok	217	128	0.597	24.02	0.775	0.0819	1.013	23.65	0.784	0.0817

B.6 Mixtures of several image codes

Table 12 decodes equal-weight averages of several codes and compares them with the originals of each corpus. Across classes, the FID of InterpTok's decoded averages of two codes is six times lower than MAETok's, and for averages of ten codes still slightly lower. On same-class pairs, the Inception Score (Salimans et al., 2016) of InterpTok's decoded mixtures is about the same as that of its reconstructions.

Table 12: **Image quality of decoded mixtures.** Equal-weight code mixtures on ImageNet and MS-COCO against each corpus's own 50,000 originals, $n = 50{,}000$ tuples per cell.

	ImageNet									MS-COCO
	1 code	adjacent codes, same class				codes of random classes				2, random
		k=2	3	5	10	k=2	3	5	10	
FID ↓										
MAETok	0.57	35.12	72.80	121.02	181.21	87.49	143.46	186.22	238.52	85.21
InterpTok	**0.57**	**3.79**	**6.71**	**18.59**	**76.23**	**13.78**	**40.81**	**124.85**	**219.07**	**13.61**
Inception Score ↑										
MAETok	138.1	47.7	24.5	11.4	4.9	15.8	8.0	4.5	2.4	10.6
InterpTok	**140.1**	**139.4**	**131.2**	**92.0**	**24.9**	**58.0**	**30.4**	**11.4**	**2.7**	**20.7**
sFID ↓										
MAETok	3.73	20.13	36.26	66.33	130.75	23.11	41.34	77.95	160.41	27.39
InterpTok	**3.25**	**10.05**	**14.57**	**19.83**	**47.17**	**13.92**	**21.81**	**30.44**	**85.44**	**15.65**
Precision ↑										
MAETok	0.999	0.442	0.294	0.228	0.130	0.187	0.141	0.134	0.048	0.189
InterpTok	**1.000**	**0.917**	**0.859**	**0.664**	**0.350**	**0.510**	**0.386**	**0.248**	**0.273**	**0.535**
Recall ↑										
MAETok	0.999	0.562	0.339	0.127	0.008	0.572	0.177	0.014	0.000	0.397
InterpTok	**0.999**	**0.612**	**0.350**	**0.157**	**0.047**	**0.787**	**0.636**	**0.323**	**0.014**	**0.697**

B.7 Decoded paths

A path can look natural at every position and still fade slowly from one image into the other, or change abruptly near the middle. Tables 13 and 14 distinguish these two cases from a path that changes steadily and looks natural throughout. Table 13 gives a path length after Karras et al. (2019), here the LPIPS between consecutive decoded images summed over nine positions. Its d_{mid} is the distance from the decoded midpoint to a source reconstruction as a share of the distance between the two source reconstructions, zero for a path that copies a source at the middle. Its ordered column is the share of pairs whose decoded images move away from the first source and toward the second at every one of the nine positions. Table 14 scores image quality and progress along the path. CMMD is the distance of Jayasumana et al. (2024) in CLIP (Radford et al., 2021) features, in thousandths. The blur column gives a path's distance to held-out real images as the strength of the Gaussian blur that moves real images equally far, lower being better. Coverage is the share of the line between the two source reconstructions that the path traverses, normalized by a source copy. The last columns of panels c and d mark whether a path has a lower blur than its own autoencoder crossfade, with a paired bootstrap interval. They also mark whether the path traverses more than half of the line, and whether both hold. InterpTok's paths reach a lower CMMD and blur than those of MAETok and SD-VAE and than every crossfade, Table 14, and nearly all of them move steadily from the first source to the second, Table 13.

Table 13: **Movement along decoded paths.** Path length in the spirit of Karras et al. (2019) with step shares and source distances, over nine equally spaced positions of the straight path, $n = 5{,}000$ per split.

	near pairs				random pairs			
Path	PPL	share	d_{mid}	ordered ↑	PPL	share	d_{mid}	ordered ↑
SD-VAE	0.796	0.174	0.739	61.4%	0.899	0.186	0.722	58.4%
MAETok	0.665	0.167	0.623	99.8%	0.799	0.172	0.636	99.4%
InterpTok	0.779	0.197	0.603	99.1%	1.056	0.203	0.628	94.8%

	near pairs		random pairs	
Reference path	PPL	share	PPL	share
pixel crossfade	0.433	0.198	0.530	0.215
endpoint copy, $d_{\text{mid}} = 0$	0.671	1.0	0.768	1.0

Table 14: **Image quality and progress along paths**, $n = 25{,}000$ paths per row.

(a) Cross-class pairs

Path	CMMD ↓	blur ↓	coverage ↑	over half
InterpTok				
line in the latent space	**171.3**	**0.547**	**0.637**	✓
autoencoder crossfade	462.4	0.876	0.525	✓
MAETok				
line in the latent space	508.5	0.945	0.613	✓
autoencoder crossfade	493.4	0.922	0.530	✓
SD-VAE				
line in the latent space	857.3	1.520	0.622	✓
autoencoder crossfade	598.4	1.088	0.540	✓
Pixel crossfades, source copies and constant outputs				
pixel crossfade	368.3	0.740	0.452	×
copy A	−0.1	0.250	−1.200	×
hard cut	−0.1	0.250	−0.200	×
grey	5053.6	>8	−0.058	×
noise	6653.4	>8	−0.038	×
endpoint snap (ours)	58.2	0.368	−0.943	×
collapse (ours)	58.7	0.369	−0.106	×

(b) Reconstruction ceiling

Tokenizer	CMMD ↓	blur ↓
InterpTok	57.7	0.367
MAETok	72.9	0.397
SD-VAE	98.9	0.450

(c) Against the autoencoder crossfade, cross-class pairs

Tokenizer	$\Delta\sigma$ [interval]	closer
InterpTok	+0.295 [0.285, 0.305]	✓
MAETok	−0.015 [−0.024, −0.005]	×
SD-VAE	−0.265 [−0.276, −0.255]	×

(d) Against the autoencoder crossfade, same-class pairs

Tokenizer	$\Delta\sigma$ [interval]	units	closer	over half	both
InterpTok	+0.287 [0.280, 0.295]	7936	✓	✓	✓
MAETok	+0.012 [0.004, 0.020]	8060	✓	✓	✓
SD-VAE	−0.289 [−0.298, −0.279]	7965	×	✓	×

B.8 Interpolation position and geometry

We also decode other positions on the line and a spherical path, Table 15. InterpTok's advantage over MAETok is largest near the middle of the line, where decoding is hardest. On a spherical path, which White (2016) recommends for Gaussian latent spaces, InterpTok's midpoint FID is even lower than on the straight line. Arvanitidis et al. (2018) also show that the straight line is not the shortest path in a curved latent space. Measured by $d_A/(d_A + d_B)$ in pixels and in LPIPS on 128 pairs, both tokenizers place their decoded images at similar positions between the sources. Diffusion methods that interpolate between two images or two text prompts change the generator or its sampling, not the tokenizer (Wang & Golland, 2023; Zhang et al., 2024; Yang et al., 2024; Zheng et al., 2024; He et al., 2024; Guo et al., 2024).

Table 15: **Position along the path and interpolation geometry**, FID on $n = 50{,}000$ decoded images per cell.

(a) Position beyond the midpoint

Position	MAETok	InterpTok (41k)
$\alpha = 0.625$	10.17	5.35
$\alpha = 0.75$	6.12	5.62
$\alpha = 0.875$	4.76	5.25

(b) Pairing and path

Pairing and path	MAETok	InterpTok (41k)	InterpTok (82k)
Near, straight	17.28	5.58	5.82
Near, spherical	17.15	4.27	4.38

B.9 Why generation quality changes with the sampling steps, the sampler and the training objective

Observation. With fewer sampling steps, InterpTok's generator reaches a lower gFID, while the gFID of the generator on MAETok's codes rises with every reduction of the steps, Table 2.

Hypothesis. A conditioned generator has two cues for the clean code it should produce, its condition, here the class label, and the noisy code as it currently looks. In each sampling step, it predicts the clean code from both cues, and the sampler moves the noisy code part of the way toward this prediction. At high noise, the noisy code carries almost no signal, and the label is the only real cue. The prediction that minimizes the denoising loss on the training data is an average of the training codes, each weighted by how well it matches the noisy code. With much noise, the noisy code is

about equally similar to every training code, every code receives about the same weight, and the prediction approaches the plain average of all training codes (Karras et al., 2022). Xu et al. (2026) show the same for the codes of an image tokenizer. For a generator conditioned on a class label, the average runs over the training codes of that class, and its prediction at high noise therefore lies near the average of these codes, which we call the class average. As the noise falls, patterns appear in the noisy code, coarse ones first and fine ones last (Skorokhodov et al., 2025), and the prediction moves toward the average of the training codes that match these patterns. The lower the noise, the more the patterns steer the prediction. Generated samples therefore tend to lie between the modes of the training data (Aithal et al., 2024). With fewer sampling steps, each step is larger, and the generator has fewer steps in which to follow the patterns that emerge in the code. It moves in larger steps toward the class average of its label and ends up closer to that average before the patterns can steer it away. Whether this helps depends on the tokenizer, since the encoder decides which codes lie around a class average and the decoder determines how the code is decoded into an image. To our knowledge, no previous work connects the averaging of a denoising generator, and the shift from its condition to the patterns that emerge in the code, to the number of sampling steps and to what the tokenizer makes of the resulting average.

Testable consequences. The hypothesis explains the observation if five things hold, and each of them can be measured. First, at high noise, the generator's prediction lies near the class average of its label, which shows in the distance of its first prediction from the class average. The label of the early steps then also sets the class of the final image, which shows when the label is changed during sampling. Second, with fewer steps, the generated codes of a class end closer to their class average and thus closer to each other, which shows in their distance from their mean. Third, with InterpTok's encoder, the codes around a class average belong to that class more often than with MAETok's encoder, which shows when we count the codes of a class among the codes nearest to its class average. Fourth, InterpTok's decoder decodes averages of codes into more natural images than MAETok's decoder, which shows when we decode averages of several codes of one class. Fifth, with fewer steps, the images of InterpTok's generator show their class more often, which a classifier can measure. We compute all class averages from the codes of the ImageNet validation images, fifty per class.

The label at high noise. We measure the distance of the generator's prediction at the first of 250 steps from the class average of its label and from the average of all validation codes, Table 18. For both generators, the first prediction lies three to four times closer to the class average of its label than to the average of all validation codes. At high noise, the prediction thus follows the label, as the hypothesis expects. We then condition the generator on one label for the first steps of a run of 250 steps and on a second label for all remaining steps. A classifier trained on ImageNet assigns each final image to a class, Table 16. With InterpTok's generator, the first label for only the first fifth of the steps already yields three quarters as many images of the first class as the first label for all steps, and almost no images of the second class. With the generator on MAETok's codes, the first label for the first fifth of the steps yields only a tenth as many images of the first class as the first label for all steps. Its images no longer show the second class once the first label conditions three fifths of the steps. For both generators, the label of the earlier, noisier steps thus sets the class of the final image, although the second label conditions all later steps. Studies of unconditional diffusion models also find that the class of an image forms at high noise and its details at lower noise (Biroli et al., 2024; Sclocchi et al., 2025).

Fewer sampling steps. The sampler of Table 2 adds fresh noise to the code after every step. The generator then predicts the clean code from this noisier code, and its prediction replaces part of the fine details it produced in the earlier steps with an average over the training codes that match the noisier code. This is intended to correct errors made in the earlier steps (Karras et al., 2022; Song et al., 2021; Xu et al., 2023). However, we argue that these so-called errors come from the generator deviating early from the condition cue, because it believes it recognizes patterns in the code. With added noise, the generator ends closer to averages over training codes than without the noise. By the hypothesis, fewer steps also make the generator end closer to an average over training codes, the class average of its label, and with this sampler the two effects cannot be told apart. We therefore sample without noise, so that only the number of steps changes. We sample the generators on the codes of MAETok, of InterpTok and of its two variants trained without interpolation and without the regularization losses, each with 250 steps and with fewer steps down to 8. For each, we measure the average distance of the generated codes of a class from their mean, Table 20. Compared with 250 steps, this distance shrinks by about a twentieth with 32 steps, a tenth with 16 and a fifth with 8, for every generator alike. Fewer steps thus bring the generated codes of a class closer together for every

Table 16: **Two class labels in one run.** The generator is conditioned on one class label for the first steps of a run of 250 and on a different label for every step after them. Shares in percent, $n = 10{,}112$ per column, the sampler that adds no noise. Above, the share of generated codes whose nearest class average is that of the first or of the second class. Below, the share of decoded images that a ResNet-50 trained on ImageNet assigns to each class.

Steps under the first class	0	10	25	50	150	250
codes nearest the average of the first class, percent						
generator on MAETok's codes	0.1	0.1	0.7	9.4	37.0	38.6
InterpTok's generator	0.0	0.5	29.0	65.8	70.8	71.2
codes nearest the average of the second class, percent						
generator on MAETok's codes	38.6	31.3	16.5	2.7	0.0	0.0
InterpTok's generator	69.9	46.0	5.9	0.1	0.0	0.0
images assigned to the first class, percent						
generator on MAETok's codes	0.0		–	5.8	49.6	59.3
InterpTok's generator	0.0		–	44.5	58.2	60.9
images assigned to the second class, percent						
generator on MAETok's codes	59.8		–	8.7	0.1	0.0
InterpTok's generator	60.5		–	0.1	0.0	0.0

generator, as the hypothesis expects. With the sampler of Table 2, the distance also shrinks with fewer steps, and with 250 steps it is about one percent shorter than without the noise. Since the distance shrinks alike for every generator, the number of steps alone does not explain why InterpTok's gFID falls with fewer steps while MAETok's rises.

The encoder. A generated code near a class average resembles an average over training codes of that class, since the label restricts the prediction to that class. What the decoder makes of such a code depends on the codes around it, since the decoder was trained only on the codes of training images and knows no labels. Which codes lie around a class average depends on the encoder. We count how many of the fifty validation codes of a class lie among the hundred validation codes nearest to its class average, Table 21. With InterpTok's encoder, about three quarters of them lie there, twice as many as with MAETok's encoder, and a code's nearest neighbour belongs to its own class twice as often as with MAETok's encoder. The ten validation codes nearest to a generated code also carry the label of the generated code three times as often with InterpTok's generator as with the generator on MAETok's codes, Table 20. Around a class average and around a generated code, InterpTok's encoder thus places more codes of the label's class than MAETok's encoder, as the hypothesis expects. In the terms of Sec. 3.2, InterpTok's encoder has more learnability than MAETok's. Most of this difference comes from the regularization losses. The variant trained without interpolation places codes of a class around its class average about as often as InterpTok, and the variant trained without the regularization losses least often of the four encoders, Table 21.

The decoder. A code near a class average resembles an average of codes, and a decoder trained only on the codes of training images was never trained on such averages. We decode equal-weight averages of two, three, five and ten neighbouring codes of one class and measure the Inception Score (Salimans et al., 2016) of the decoded images, which rises when a classifier recognizes the object in each image with more confidence, Table 12. For two codes, InterpTok's decoder keeps the Inception Score of a single decoded image, while MAETok's decoder falls to a third of it. For three and five codes, InterpTok's decoder keeps nine tenths and two thirds of it, while MAETok's decoder keeps a sixth and a twelfth, and for ten codes both fall far below it. For averages of two codes of one class, the FID of the decoded images is six times lower with InterpTok's decoder than with MAETok's, also when our decoder decodes MAETok's codes. The decoder of the variant trained without interpolation stays as high as MAETok's, Table 19. On MAETok's codes, our decoder also changes the decoded image about a fifth less than MAETok's decoder when a code is displaced a short distance toward its nearest neighbour, Table 22. A decoder trained between codes thus decodes averages of codes into more natural images than a decoder trained only on the codes of training images, which decodes them into blends with artifacts. In the terms of Sec. 3.2, InterpTok's decoder has more decodability than MAETok's. This agrees with Xu et al. (2026), who find across thirteen tokenizers that what a decoder produces between training codes predicts generation quality, while reconstruction quality does not.

The generated images. We decode the generated codes, and a classifier trained on ImageNet assigns each image to a class, Table 23. Without the sampler's noise, the share of InterpTok's images that show the class of their label rises by a third from 250 to 8 steps, while it stays flat for the generator on MAETok's codes with MAETok's decoder. With our decoder, the images decoded from MAETok's generated codes also show their class more often with fewer steps, but their share rises by less than

half as much as InterpTok's. With the sampler's noise, the share rises for InterpTok from 250 to 32 steps and stays flat on MAETok's codes with either decoder. With fewer steps, InterpTok's generated images thus show their class more often, as the hypothesis expects.

Summary. We measured that the generator's first prediction lies much closer to the class average of its label than to the average of all codes, and that the label of the early, noisier steps sets the class of the final image. With fewer steps, the generated codes of a class end closer together, for every generator alike. InterpTok's encoder places more codes of the label's class around a class average than MAETok's encoder, and InterpTok's decoder decodes averages of codes into more natural images than MAETok's decoder. With fewer steps, InterpTok's images show their class more often, while those of the generator on MAETok's codes do not. These measurements support our argument, and we hypothesize that a conditioned generator first follows its label toward the average code of the class, and then the patterns that emerge in the code toward the training codes that match them. Under this hypothesis, fewer steps give the generator fewer chances to follow the patterns, and it ends closer to the class average than with many steps. Intuitively, with many steps the generator produces a particular member of the class, and with few steps a more generic one. Whether a more generic code becomes a natural image of the class depends on the tokenizer. With InterpTok, codes of the class surround the class average more often than with MAETok, and InterpTok's decoder turns the average into a more natural image than MAETok's decoder. With MAETok, more codes of other classes surround the class average than with InterpTok, and its decoder turns the average into a blend.

The five tokenizers. FID compares the feature distribution of generated images with that of real images (Heusel et al., 2017). It is therefore lower when the generated images look more like real images and when each class appears as often as in the dataset. With fewer steps, every generator ends closer to the class averages of its labels. Whether this lowers the gFID depends on learnability, which decides how often the codes around a class average belong to that class, and on decodability, which decides how natural an image the decoder makes of a code there. With the sampler of Table 2, fewer steps also weaken the correction by the noise. With 250 steps, the generated codes end about one percent closer to their nearest validation code and to each other with the noise than without it, and with 32 steps no longer, Table 20. A sampler that adds noise also needs more steps to reach the same quality than one that adds none (Ma et al., 2024; Xu et al., 2023). We hypothesize that with this sampler, the gFID falls with fewer steps only when the gain from the class average outweighs the loss of this correction.

Without the sampler's noise, there is no correction to lose, and the gFID falls with fewer steps for every tokenizer that has more learnability or more decodability than MAETok, Table 17. From 250 to 16 steps, it falls by a fifth for the variant trained without interpolation, which has more learnability than MAETok. It falls by a fifth as well for the variant trained without the regularization losses, which has more decodability than MAETok. From 250 to 8 steps, it falls by nearly half for MAETok's codes with our decoder, which has more decodability than MAETok, and by three fifths for InterpTok, which has more of both. For MAETok, it changes by less than two percent from 250 to 16 steps.

With the sampler of Table 2, the gFID of MAETok rises with fewer steps. Around its class averages lie more codes of other classes than around InterpTok's, and its decoder decodes the averages into blends. Fewer steps thus bring no gain that could outweigh the weaker correction.

With our decoder, MAETok's generated codes reach a lower gFID than with MAETok's decoder at every number of steps, since our decoder decodes averages of codes into more natural images, Table 17. With 250 steps, a decoder fine-tuned with only the interpolation discriminator gives more than half of this gain, and a decoder fine-tuned with MAETok's own losses about a quarter. The gFID still rises with fewer steps, since the codes around MAETok's class averages belong to other classes more often than with InterpTok's encoder, and the gain from the class average stays smaller than for InterpTok. With 250 steps, it nearly matches InterpTok. We hypothesize that with many steps, the generated code follows the patterns toward the training codes nearest to it, and the class structure of the latent space matters less than with few steps.

The variant trained without interpolation has about the learnability of InterpTok, but its decoder decodes averages of codes with about the same FID as MAETok's decoder, Table 19. Its generated codes lie among codes of their class even more often than InterpTok's, Table 20, but its images show their class less often than MAETok's, Table 23. Its learnability without decodability thus does not make its images show their class more often. Its gFID is the highest of all variants, and it rises with fewer steps.

The variant trained without the regularization losses has about the decodability of InterpTok, since its

interpolation quality is close to InterpTok's, Table 3, but the least learnability of the four encoders. Around its class averages lie mostly codes of other classes, Table 21. The ten validation codes nearest to one of its generated codes carry the generated code's label half as often as with the generator on MAETok's codes, at every number of steps, Table 20. Its decoder decodes these codes into more natural images than MAETok's decoder, but its images show their class less often than MAETok's, Table 23, and its gFID is higher than MAETok's and rises with fewer steps. With MAETok's encoder frozen, a decoder trained with the interpolation discriminator reaches about half the gFID of this variant, and training the encoder lowers the linear-probe accuracy by an eighth, Table 3. Most of the difference thus comes from the encoder. We attribute this to interpolation, which pulls the codes of different images closer to each other, while no loss keeps the codes of different classes apart. This variant uses none of the Inception features in which iFID is measured, and it still reaches most of the gain of InterpTok in interpolation quality.

InterpTok has both properties. With fewer steps, its generated codes end closer to class averages around which codes of the label's class lie twice as often as with MAETok's encoder, and its images show their class more often, Table 23. Its decoder decodes these averages into more natural images than MAETok's decoder, and FID rewards both. With the sampler of Table 2, the gain from the class average outweighs the weaker correction down to 32 steps, and with 16 steps the loss outweighs the gain.

We therefore hypothesize that a conditioned generator needs decodability together with learnability to gain from fewer steps. With learnability alone, as without interpolation, or with decodability alone, as without the regularization losses, the gFID is higher than MAETok's and rises with fewer steps. InterpTok, which combines both, reaches the lowest gFID and lowers it further with fewer steps.

Table 17: **Generation quality with and without the sampler's noise.** gFID of the class-conditional SiT-B generators of Sec. 4.1 and of the released SiT-XL against the ADM statistics of ImageNet 256 without guidance, $n = 50{,}000$ per cell. Without interp. and without reg. are the variants of Table 3 trained without interpolation and without the regularization losses, and + marks a decoder placed on the codes of the row above.

(a) sampler with noise

gFID ↓	250	64	32	16
SiT-B on MAETok's codes, one generator				
MAETok	11.61	12.33	13.88	20.56
+ our decoder	8.56	8.99	9.92	15.33
+ interp. discriminator only	9.71	10.16	11.26	16.92
+ reconstruction objective only	10.80		–	
SiT-B on each tokenizer's own codes				
InterpTok	8.48	8.09	8.11	12.48
without reg.	18.90	19.68	20.99	27.79
released SiT-XL on MAETok's codes, one generator				
MAETok	2.46	2.78	3.46	7.52
+ our decoder	2.26	2.45	2.79	5.80

gFID ↓, sampler with noise	250	64	32
without interp., on its own codes	21.15	21.53	22.98

(b) sampler without noise

gFID ↓	250	64	32	16	8
MAETok	15.74	15.43	15.52	16.01	19.41
+ our decoder	13.25	–	11.47	9.82	7.50
InterpTok	16.17	14.02	11.49	7.93	6.38

gFID ↓, sampler without noise	250	32	16
without interp.	29.14	24.98	22.64
without reg.	23.09	20.68	18.05

Table 18: **The generator's estimate at the first and the last sampling step.** Mean distance from the estimate of the noise-free code to the reference in each column, in the normalized latent space of the generator. Sampler without noise at 250 steps, 50,000 starts per generator.

Step	Noise	Own average	Other average	All codes	Final code
the generator on MAETok's codes					
1	1.00	5.3	7.8	15.6	26.3
250	0.004	27.2	26.7	31.3	0.0
InterpTok's generator					
1	1.00	10.6	24.3	39.7	44.1
250	0.004	46.2	49.7	61.9	0.0

Table 19: **Decoded averages of two validation codes.** Equal-weight averages decoded by the named decoder, $n = 50{,}000$ per cell. Each row averages the codes of its own tokenizer, and the indented row averages MAETok's codes and changes only the decoder. Class accuracy is the share of the decoded images that a ResNet-50 trained on ImageNet assigns to the class of the first code.

	Same class		Random classes	
Decoder	FID ↓	Class acc. ↑	FID ↓	Class acc. ↑
MAETok	44.6	50	101.0	8
+ our decoder	7.7	81	29.5	29
without interp.	43.3	50	93.8	10
InterpTok	**7.4**	**85**	**20.7**	**36**

Table 20: **Generated codes at four numbers of sampling steps.** 50,000 generated codes per generator, sampler without noise unless the block says otherwise. Block d is the distance of the generated code at 8 steps from the generated code at 250 steps from the same starting noise, in percent of the length of the code.

Sampling steps	250	32	16	8
a. within-class spread, relative to 250 steps				
MAETok	1	0.96	0.92	0.84
InterpTok	1	0.95	0.90	0.79
without interp.	1	0.95	0.89	0.78
without reg.	1	0.96	0.91	0.81
b. share of the ten nearest validation codes that carry the code's label, percent				
MAETok	18	19	20	21
InterpTok	55	58	61	66
without interp.	59	62	65	72
without reg.	10	10	10	10
c. distance to the nearest validation code, relative to the spacing of validation codes				
MAETok	1.00	0.98	0.96	0.92
InterpTok	0.99	0.97	0.95	0.90
without interp.	1.01	0.99	0.96	0.92
without reg.	1.00	0.98	0.95	0.89
d. distance of the code at 8 steps from the code at 250 steps, percent of the length of the code				
MAETok				13
InterpTok				26
without interp.				23
without reg.				10

Sampling steps	250	32
e. within-class spread with the sampler with noise, relative to 250 steps without noise		
MAETok	0.99	0.97
InterpTok	0.99	0.95
f. distance to the nearest validation code with the sampler with noise, relative		
MAETok	0.99	0.98
InterpTok	0.98	0.97
g. distance to the nearest segment between two of the ten nearest validation codes, relative to the distance to the nearest one		
MAETok	0.84	0.84
InterpTok	0.83	0.83

Codes that are not generated	MAETok	InterpTok
h. as in block g, where moved at random means a validation code moved in a random direction by a generated code's distance to its nearest validation code		
a validation code	0.85	0.83
moved at random	0.93	0.92
i. as in block c, absolute		
a validation code	31	50
moved at random	44	71

Table 21: **The latent space of each encoder.** The codes of the 50,000 ImageNet validation images, in the normalized latent space of the generator unless a block says otherwise. The linear probe follows the MAE procedure of Table 27. Without interp. and without reg. are the variants of Table 3. †Quoted from Zheng et al. (2026) under its own procedure.

	MAETok	InterpTok	without interp.	without reg.
Linear-probe accuracy, percent	73.0	77.9	77.0	63.5
Nearest neighbour of the same class, percent	30	61	65	17
Nearest-neighbour distance, relative	1	1.6	1.6	0.9
Own-class codes among the 100 nearest to a class average, of at most 50	17	37	39	5
Nearest class average is its own, percent	55	73	73	43
in the unnormalized latent space of the decoder				
Per-dimension variation, relative	1	2.2	2.1	0.8
Effective dimensions, the participation ratio of the covariance spectrum, relative	1	2.1	3.1	1.2

	MAETok	InterpTok
Between-class to within-class spread, relative	1	2

Two further encoders	Linear probe
SD-VAE	4.9
RAE†	84.5

Table 22: **Decoded images after a fixed displacement of the code.** 5,000 validation codes, five per class, moved by the distance in the header and decoded.

Distance	2	4	8	16	32
a. perceptual distance, moved toward the nearest validation code					
MAETok	0.018	0.060	0.180	0.429	0.666
+ our decoder	0.015	0.048	0.140	0.428	0.660
without interp.	0.005	0.017	0.058	0.174	0.424
InterpTok	0.003	0.012	0.036	0.106	0.318
b. perceptual distance, moved in a random direction					
MAETok	0.038	0.090	0.189	0.381	0.616
+ our decoder	0.037	0.091	0.206	0.419	0.632
without interp.	0.073	0.170	0.374	0.646	0.846
InterpTok	0.054	0.135	0.319	0.553	0.700
c. class kept, percent, moved toward the nearest validation code					
MAETok	77	76	68	35	29
+ our decoder	77	77	75	43	29
without interp.	76	76	75	71	56
InterpTok	77	77	77	76	71
d. class kept, percent, moved in a random direction					
MAETok	77	75	68	23	1
+ our decoder	77	74	63	9	0
without interp.	75	67	19	0	0
InterpTok	77	74	57	10	1

	MAETok	without interp.	InterpTok
Distance to the nearest validation code, for scale	29	63	77

Table 23: **Class accuracy and similarity of the decoded images**, 50,000 decoded images per cell. Class acc. is the share of the decoded images whose top-1 class under a ResNet-50 trained on ImageNet is the conditioned label, and sim. is the mean cosine similarity of the features of those images within a class.

	Class acc., percent				Sim.			
Sampling steps	250	32	16	8	250	32	16	8
MAETok	60	60	61	58	0.42	0.43	0.43	0.45
+ our decoder	62	64	66	71	0.42	0.43	0.44	0.47
InterpTok	61	67	73	83	0.43	0.45	0.48	0.56

	Class acc., percent				Sim.			
Sampling steps	250	32	16	8	250	32	16	8
without interp.	49	53	56	57	0.41	0.43	0.45	0.48
without reg.	50	52	55	61	0.40	0.41	0.42	0.45

	Class acc., percent		Sim.	
Sampling steps	250	32	250	32
with the sampler's noise, the draws of Table 2, whose generation quality they reproduce within 0.2				
MAETok	64	63	0.43	0.43
+ our decoder	67	66	0.43	0.44
InterpTok	68	71	0.44	0.47

C INTERPOLATION UNDER A SMALLER CODE

We ran an earlier experiment with a much smaller code and record what it produced beside the main result.

The tokenizer is a frozen REPA-E end-to-end autoencoder (Leng et al., 2025) read by a transformer encoder at SiT-B scale and written back by a flow decoder at SiT-L scale (Ma et al., 2024). Only those two are trained, and they carry the code through thirty two register tokens of thirty two dimensions each, normalised together onto a sphere. Some other implementation details differ from InterpTok as well.

The smaller code is more semantic, and its mixtures change the content of the scene more and superimpose the two images less, a mixture of a beach and a cliff being one coastline. Fig. 21 shows the decoded images at positions along the line between two image codes, and Fig. 22 decodes mixtures of four codes placed at the corners of a square.

Reconstruction is the cost, and Table 24 sets it beside MAETok and InterpTok.

Table 24: **Reconstruction under a smaller code.** ImageNet, 50,000 images, SSIM in MAETok's convention.

Tokenizer	PSNR ↑	SSIM ↑	LPIPS ↓
InterpTok	24.02	0.775	0.0819
MAETok	23.55	0.760	0.0917
Smaller code	19.31	0.615	0.1735

One variant sampled random codes alongside the interpolated ones, since a smaller and more semantic code should make that easier, and it collapsed, in the way a generative adversarial network does, onto a single incoherent image, decoding a meaningful image from an arbitrary code being a more ambiguous task than decoding one between two image codes.

A smaller bottleneck can make the codes more semantic at the cost of reconstruction, and for interpolation quality, and for semantic interpolation in particular, it can be the better choice.



Figure 21: **Decoded paths under a smaller code.** The two dark outlined ends decode the codes of two images generated with Z-Image-Turbo (Z-Image Team et al., 2025) and the seven tiles between them mixtures of those two codes.



Figure 22: **Decoded mixtures of four codes under a smaller code.** Four unrelated squares, one per quadrant, each tile decoded from a mixture of its four corner codes with weights bilinear in the position. Only the dark outlined corners hold images generated with Z-Image-Turbo (Z-Image Team et al., 2025).

D Formal Observations

Preliminary. Let E and D be the encoder and decoder, $\mathcal{X}$ a finite set of training images, and $S = E(\mathcal{X})$ its codes in $\mathbb{R}^{128\times32}$. For sources A and B with codes $a = E(A)$ and $b = E(B)$, the mixture $ab_\alpha = (1-\alpha)a + \alpha b$ traces the segment between them for $\alpha \in [0, 1]$.

Assumptions. For Gaussian corruption, let γ_t be the signal scale and σ_t the noise standard deviation. For Theorem 3 of Xu et al. (2026), assume bounded codes and a positive gap between the squared distances to the second and third nearest codes in $\gamma_t S$. Take $\sigma_t \to 0$ and $\gamma_t \to 1$ while this gap stays bounded away from zero. For the local dynamics of He et al. (2026), assume bounded, separated observations and a smooth ridge with positive reach, so that nearby points have a unique projection onto it. Assume the global and normal posterior-mean errors in their concentration bounds are small.

D.1 Reconstruction and interpolation

A reconstruction objective is a sum over the codes of training images, and it constrains a decoder at those codes only. The three observations below state which measurements distinguish two decoders under this constraint and which do not.

Observation D.1 *Fix E, finitely many masked views $\mathcal{V}$ and a finite pair schedule $\mathcal{P}$. Assume at least one scheduled pair has distinct codes, and let*

$$S_{\mathrm{rec}} = E(\mathcal{X} \cup \mathcal{V}), \qquad S_{\mathrm{int}} = \bigcup_{(A,B)\in\mathcal{P}} \{ab_\alpha \mid \alpha \in (0,1)\}.$$

For an empirical reconstruction objective evaluated on S_{rec} and any decoder perturbation h,

$$h|_{S_{\mathrm{rec}}} = 0 \quad \Longrightarrow \quad L_{\mathrm{rec}}[D + \epsilon h] = L_{\mathrm{rec}}[D].$$

Proof. Every term of the reconstruction objective evaluates the decoder at a code in S_{rec}, where h is zero. Since S_{rec} is finite and every scheduled segment holds infinitely many codes, h can still be nonzero on most of S_{int}. The observation concerns where the objective is evaluated. The decoder's parameters are shared, and an update at those codes also changes the decoder elsewhere. Sec. 4.2 measures how far a trained decoder generalizes from them.

Observation D.2 *Fix labelled data, its codes and all randomness in the evaluation procedures. Decoders that agree at every evaluated code share rFID, PSNR, SSIM and every diagnostic computed only from those codes and labels, including linear-probe and nearest-neighbour accuracy. Their interpolated FIDs can differ, since this agreement leaves the images between the evaluated codes unconstrained.*

Proof. Reconstruction diagnostics receive identical images, and code diagnostics receive identical codes and labels.

Observation D.3 *Let E, D be a deterministic continuous tokenizer and $T : \mathbb{R}^d \to \mathbb{R}^d$ a continuous bijection with continuous inverse. For $E' = T \circ E$ and $D' = D \circ T^{-1}$, reconstruction and fixed-pair interpolation satisfy*

$$D'(E'(X)) = D(E(X)),$$
$$D'((1-\alpha)E'(A) + \alpha E'(B)) = D\big(T^{-1}((1-\alpha)T(a) + \alpha T(b))\big).$$

Proof. The first identity cancels $T^{-1} \circ T$, and the second substitutes the definitions. An invertible affine T commutes with the mixture and leaves the decoded path unchanged, while a nonlinear T generally changes it. Interpolation can therefore distinguish two coordinate systems that reconstruct identically. In practice, regularizers and finite networks can favour particular coordinates, and a different distance metric pairs different neighbours.

D.2 Sampling between neighbouring codes

The three statements below restate results of others under the assumptions above.

Statement D.4. Empirical scores. Under Gaussian corruption, the empirical score is determined by a distance-weighted average of the training observations (Xu et al., 2026). Reversing the noise exactly recovers the training distribution as the noise vanishes. To produce new samples, a generator's learned score has to depart from exact reversal. The learned score determines which codes between training codes are reached, and the decoder determines their images.

Statement D.5. The nearest-pair ridge. Under the nearest-pair assumptions above, Theorem 3 of Xu et al. (2026) bounds the error of approximating the score Jacobian with the two nearest codes

a and b. Their Theorem 4 gives the ridge of this approximation as the affine line through $\gamma_t a$ and $\gamma_t b$, which approaches the line through the original codes as $\gamma_t \to 1$. Its tangential eigenvalue peaks at the midpoint, while the normal attraction is constant along the line. The density $6\alpha(1-\alpha)$ from which we draw α, App. E.6, also emphasizes the middle.

Statement D.6. Local ridge dynamics. Under the local assumptions above, He et al. (2026) bound the concentration near a ridge by separating arrival in its neighbourhood, normal alignment and tangential motion. Their examples give a straight ridge for two observations and a curved ridge for three. Their random-feature analysis separates the error of a finite-width network from the optimization error, and with two observations, finite width keeps the samples spread along the line.

Related analyses of the space between observations. For uniformly spaced collinear observations, Chen (2026) construct a smoothed score under which denoising ends in a distribution with an atom at every observation and continuous mass between neighbouring observations. Aithal et al. (2024) relate movement between modes to samples outside the training support. These results describe the score between observations, and our measurements describe what a decoder produces there.

D.3 Interpolation and image representations

A set of N codes gives up to $O(N^2)$ segments, each with infinitely many positions, without any new image. Along a segment, the two sources and the mixing ratio specify how natural the decoded mixture should look and how much of each source it should show, without specifying a target image. The interpolation objective is therefore self-supervised, as masked modelling is, with constraints in place of a masked target. Through the discriminator and the frozen vision model, the tokenizer learns from pretrained networks on images that do not exist in the dataset. Xu et al. (2026) found no straightforward way to minimize iFID directly, and training on the line trains the decoder on the kind of image that iFID scores. Token dropout and masked prediction encourage abstract image codes (Bachmann et al., 2025; He et al., 2022; Chen et al., 2025a). We hypothesize that interpolation is a third way to make codes abstract at a fixed code size. The mixtures of several codes in App. B.6 are consistent with this, since no training pair contains those combinations. Zhong et al. (2026), Gu et al. (2026) and Yue et al. (2026) study which latent spaces a diffusion model learns more easily. Of these, Yue et al. (2026) train the neighbourhood of one code, while we train the whole segment between two codes.

E Experiments Setup

E.1 Training Details of the Tokenizer

Table 25 gives the training configuration. MAETok masks 40 to 60% of the image patches and trains three auxiliary decoders that regress HOG, DINOv2-L and SigLIP-SO400M features of the masked patches from the code. A second stage freezes the encoder and fine-tunes the pixel decoder under reconstruction, perceptual and adversarial losses (Chen et al., 2025a). Our fine-tune uses neither masking nor auxiliary decoders. The CLS feature of the frozen vision model and a list of the 64 nearest neighbours are precomputed for every training image. The variants of Sec. 4.3 are compared at a matched number of steps, except the one with only the interpolation discriminator, Table 3.

Table 25: **Tokenizer training configuration.** The ImageNet fine-tune behind every reported result.

Configuration	Value
Data and pairing	
Training images	ImageNet-1k train, 1,280,602 images at 256×256 in $[-1, 1]$
Held-out block	512 images, reserved from the fine-tune only
Pair sampling	per-epoch disjoint kNN matching, neighbour rank $k = 8,\ 8,\ 16,\ 16,\ 32,\ 32,\ 64,\ 64,\ 128,\ 128$ over epochs 1–10, kNN depth $K = 64$, neighbour rank 128 from epoch 9, about 4,500 steps per epoch
α sampler	Beta(2,2) by exact inverse CDF, one α per pair, endpoints never drawn
Tokenizer and rows	
Base model	MAETok, encoder and decoder trainable
MAETok's published schedule	500,000 steps at global batch 512
Latent	128 tokens $\times$ 32 channels, unconstrained (no KL, no quantisation)
Decoder growth	$12 \rightarrow 18$ blocks, six zero-gated copies of blocks 6–11 ($+43.1$M parameters), bit-identical to MAETok's decoder at step 0
Micro-batch	8 images $\rightarrow$ 28 pairs $\rightarrow$ 36 decoder rows in one decode call
Global batch	256 images per optimiser step (32 accumulated micro-batches), 896 mixtures and 1,152 decoded images per step
Optimisation	
Tokenizer	AdamW, lr 10^{-5}, $\beta = (0.9, 0.99)$, weight decay 0.01, constant schedule, grad clip 1.0
Discriminator	AdamW, lr 10^{-4}, $\beta = (0.9, 0.95)$, weight decay 0, grad clip 1.0
EMA	0.998, one update per optimiser step, and the EMA copy is the one every number in this paper is measured on
Precision	bf16 autocast, LPIPS in fp32
Seed and checkpoints	seed 0, one checkpoint every 1,000 steps, all kept
Compute	
Hardware	8$\times$ A100-40GB, one node
Wall and steps	72 h to step 41,000, 6.15 s per step late in the run
Peak memory	33.01 GB per GPU
Resumed training	resumed from step 41,000, a further 72 h to step 82,000

Table 26: **Discriminator, vision model and generator settings.** The frozen backbones act during training only.

Configuration	Value
Discriminator	
Discriminator backbone	frozen DINO ViT-S/16 at 224 px, 5 taps, 196 tokens per tap
Discriminator heads	the interpolation discriminator's head, which scores the decoded mixture beside one source, and the unconditional reconstruction discriminator's head, both on the shared backbone
Discriminator parameters	25.86M trainable (2$\times$ 7.39M of heads and 11.09M endpoint cross-attention); backbone frozen at 21.67M
Adversarial schedule	discriminator from step 0 with a 1,500-step D-only warm-up, D:G 1:1, adaptive weight capped at 1.0
Adaptive weight	the VQGAN ratio $\|\nabla\mathcal{L}_{\text{rec}}\|/(\|\nabla\mathcal{L}_G\| + 10^{-4})$ at the decoder's final linear layer, clamped to $[0, 10^4]$ as upstream and then capped at 1.0
Discriminator regularisers	LeCAM 10^{-3} at EMA decay 0.999, consistency regularisation 4.0 outside the 0.4 weight, DiffAug colour, translation and cutout at probability 0.5 on both networks, and no gradient penalty
Head stack per tap	spectral-normalised (Miyato et al., 2018) 1-d convolutions, a k=1 projection, local batch normalisation and LeakyReLU at slope 0.2, a residual k=9 circular convolution block scaled by $1/\sqrt{2}$, and a k=1 projection to one logit per token
Endpoint cross-attention	pre-layer-normalised spectral-normalised linear maps, 6 heads, scaled dot-product attention, a position-wise MLP at 4$\times$ expansion with GELU, LayerScale initialised at 10^{-4} on both residual branches so the block is the identity at initialisation, and a k=1 spectral-normalised convolution fusing $[x;\ \texttt{att};\ x - \texttt{att}]$
Passes per mixture	four, the score belonging to the query, with the two real and the two fake passes averaged before the discriminator hinge and all four carrying generator gradient
Representation Fréchet loss	
Fréchet moments	bias-corrected EMA at decay 0.999 of the all-reduced global-batch moments, blended with the current batch at weight $1 - \beta$, with an eigenvalue matrix root in fp64
Fréchet warm-up	accumulate only, the term enters the loss after 500 optimiser boundaries
Frozen vision model	
Backbone	frozen DINOv3 ViT-B/16 pretrained on LVD-1689M, CLS and patch features precomputed for every training image
Losses	distance-matrix alignment 0.5, latent-CLS alignment 0.25 and decoder-patch alignment 0.1 on single images, the feature mixing ratio loss 0.5 on at most 8 of the 28 mixtures of a micro-batch
Generator	
Model	SiT-B/1D (Ma et al., 2024) on each tokenizer's codes, one generator per tokenizer
Training steps	400,000 steps in the pipeline released with the iFID paper
Seed	seed 0, one training run per tokenizer
Sampling and evaluation	
Sampler	SDE at 250 steps, classifier-free guidance scale 1.0 (no guidance)
Samples	50,000 class-conditional samples per tokenizer
Metric	FID-50k against the ADM reference statistics for ImageNet 256×256, compared with our own MAETok control, App. B.1

E.2 Training Objectives

A micro-batch is the part of a batch that one forward pass processes. It contains n single images and m mixtures, as many as Table 25 states, and every pair of its single images gives one mixture. Every fine-tuning step therefore trains reconstruction at the codes of training images and the interpolation objective between them, with three and a half times as many mixtures as single images. Each loss below is listed with its formula, its weight at the start of training and the images it acts on, single images or mixtures. For an image X, $D(E(X))$ is its reconstruction, ab_α and AB_α are as in Sec. 3, and f_A, f_B and f_{AB} are the unit CLS features of A, B and AB_α in the frozen vision model, DINOv3-B/16.

1. **Reconstruction** $\mathcal{L}_{\text{rec}} = \frac{1}{n}\sum_X \big(\|D(E(X)) - X\|_1 + \text{LPIPS}_{\text{VGG}}(D(E(X)), X)\big)$ (Zhang et al., 2018) acts on single images at weights 1.0 and 1.0 and is the numerator of the adaptive adversarial weight.
2. **Generator adversarial** $\mathcal{L}_G = \mathcal{L}_G^{\text{int}} + 1.0 \cdot \mathcal{L}_G^{\text{rec}}$ acts on both at weight $w = w_{\text{ad}} \cdot 0.4 \cdot g_{\text{gate}}$, with $\mathcal{L}_G^{\text{int}}$ of Eq. 7, $\mathcal{L}_G^{\text{rec}} = -\mathbb{E}[\text{logit}]$, $w_{\text{ad}} = \|\nabla\mathcal{L}_{\text{rec}}\|/(\|\nabla\mathcal{L}_G\| + 10^{-4})$ taken at the decoder's final linear layer and capped at 1.0, and $g_{\text{gate}} = 0$ for the first 1,500 steps. The alignment losses do not enter this ratio. The discriminators train from step 0, where MAETok starts its discriminator at step 30,000.
3. **Discriminator** $\mathcal{L}_D = 0.4\,[\mathcal{L}_D^{\text{int}} + 1.0 \cdot \mathcal{L}_D^{\text{rec}} + 10^{-3}\,\mathcal{L}_{\text{LeCAM}}] + 4.0\,\mathcal{L}_{\text{CR}}$ uses $\mathcal{L}_D^{\text{int}}$ of Eq. 7 and $\mathcal{L}_D^{\text{rec}} = \frac{1}{2}\,\mathbb{E}[(1 - \text{real})_+ + (1 + \text{fake})_+]$ per logit on real images against their reconstructions. LeCAM (Tseng et al., 2021) uses EMA decay 0.999, consistency regularization (Zhang et al., 2020) compares a DiffAug view with a cutout view, and DiffAug (Zhao et al., 2020) is applied to the inputs of both networks at probability 0.5.
4. **Pixel mixing ratio loss** $\mathcal{L}_{\text{ratio}}$ of Eq. 3 acts on mixtures at weight 2.0, averaged over α, with the reconstructions $D(a)$ and $D(b)$ of the same forward pass detached. The chord form $\|AB_\alpha - ((1-\alpha)D(a) + \alpha D(b))\|$ is not used, since its unique optimum is the pixel crossfade.
5. **Feature mixing ratio loss** $\mathcal{L}_{\text{ratio}}^{\text{feat}}$ acts at weight 0.5 on at most 8 of the 28 mixtures of a micro-batch, whose decoded images pass through the frozen vision model with gradient. With $\theta(u, v) = 2\,\text{atan2}(\|u - v\|, \|u + v\|)$, exact at 0 and at π where a plain arc cosine loses precision,

$$\mathcal{L}_{\text{ratio}}^{\text{feat}} = \mathbb{E}_\alpha\Big[\big(\theta(f_{AB}, f_A) - \alpha\,\theta(f_A, f_B)\big)^2 + \big(\theta(f_{AB}, f_B) - (1-\alpha)\,\theta(f_A, f_B)\big)^2\Big], \tag{5}$$

   whose zero requires $\theta(f_{AB}, f_A) + \theta(f_{AB}, f_B) = \theta(f_A, f_B)$ and places f_{AB} on the shorter great-circle arc from f_A to f_B.
6. **Representation Fréchet loss** $\mathcal{L}_{\text{fd}} = \sum_e \text{FD}_e/(\text{sg}(\text{FD}_e) + 0.01)$ acts at weight 0.3 on an 8-image subsample of the 36 decoded images of a micro-batch, single images and mixtures, over the frozen SigLIP2-L/16 pooler, the Inception-v3 pool features and the MAE ViT-L/16 CLS feature. Inception-v3 is also the feature space of FID. The loss follows Yang et al. (2026a) at EMA decay 0.999 with an eigenvalue matrix root and a stop-gradient normalizer, on reference moments of the ImageNet training images, after the moments have accumulated over 500 optimizer steps.
7. **Distance-matrix alignment** $\mathcal{L}_{\text{axis}} = \text{MSE}(\hat{z}\hat{z}^\top, \hat{t}\,\hat{t}^\top)$ acts on single images at weight 0.5, where $\hat{z}$ holds the $n \times 4096$ row-normalized flattened codes and $\hat{t}$ the $n \times 768$ row-normalized CLS features of the frozen vision model. It covers the full 8×8 matrix, diagonal included, and adapts the distance-matrix loss of VA-VAE (Yao et al., 2025), which compares the positions within one image, to pairs of images.
8. **Latent-CLS alignment** $\mathcal{L}_{\text{pred}} = \mathbb{E}[1 - \cos(h(z), t)]$ acts on single images at weight 0.25, with h a trainable MLP $4096 \to 1024 \to 768$ with SiLU on the flattened code z and t the CLS feature of the frozen vision model, with the MLP projection and cosine objective of REPA (Yu et al., 2025) applied to the code.
9. **Decoder-patch alignment** $\mathcal{L}_{\text{patch}} = \mathbb{E}_{\text{image, pos}}[1 - \cos(h'(u), t')]$ acts on single images at weight 0.1, with u the output of decoder block 6 of 18 at each of the 256 image-token positions, h' a per-position MLP $768 \to 1024 \to 768$ and t' the 256 patch tokens of the frozen vision model without gradient, following the semantic regularization of GigaTok (Xiong et al., 2025).

The full objective of InterpTok is

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{rec}} + w\mathcal{L}_G^{\text{rec}} + \lambda_1\mathcal{L}_{\text{axis}} + \lambda_2\mathcal{L}_{\text{pred}} + \lambda_3\mathcal{L}_{\text{patch}}}_{\text{single images}} + \underbrace{w\mathcal{L}_G^{\text{int}} + \lambda_4\mathcal{L}_{\text{ratio}} + \lambda_5\mathcal{L}_{\text{ratio}}^{\text{feat}}}_{\text{mixtures}} + \underbrace{\lambda_6\mathcal{L}_{\text{fd}}}_{\text{both}}, \tag{6}$$

where w is the adaptive adversarial weight of Esser et al. (2021) and every λ is the weight given above. In the terms of Eq. 4, $\mathcal{L}_{\text{disc}}$ is the adversarial term on mixtures, and $\mathcal{L}_{\text{reg}}$ collects the three alignment losses and the representation Fréchet loss.

Variants of Table 3. The variant trained without the regularization losses starts from the released weights and keeps the images, the pair schedule, the optimizer, the batch and the schedule of the full recipe. It trains encoder and decoder, without the appended blocks, under the reconstruction objective, the interpolation discriminator and the pixel mixing ratio loss. The variant trained without interpolation trains encoder and decoder with the reconstruction objective, the three alignment losses and the representation Fréchet loss, and it uses the appended decoder blocks.

E.3 Discriminator Architecture

Backbone and heads. Both discriminators read five taps of the frozen DINO ViT-S/16 backbone (Caron et al., 2021) of MAETok's discriminator, before the first block and after blocks 2, 5, 8 and 11, with the CLS token added to every patch token. Each tap carries a spectral-normalized (Miyato et al., 2018) head with one logit per token, giving $5 \times 196 = 980$ hinge logits per image.

Conditioning and passes. The interpolation discriminator conditions each pass on one source. Before each of its heads, the tokens of the scored image query the keys and values of the other image at the same tap, and a $k{=}1$ convolution fuses $[x;\ \mathrm{att};\ x - \mathrm{att}]$, which makes their difference an explicit input. The condition thus enters through the features, as in a projection discriminator (Miyato & Koyama, 2018). With $s(X \mid Y)$ the per-token logit of image X over the keys of image Y, the relativistic comparison of Jolicoeur-Martineau (2019) uses

$$\begin{gathered} s_{\text{real}} = \tfrac{1}{2}\big[s(A \mid AB_\alpha) + s(B \mid AB_\alpha)\big], \qquad s_{\text{fake}} = \tfrac{1}{2}\big[s(AB_\alpha \mid A) + s(AB_\alpha \mid B)\big], \\ \mathcal{L}_D^{\text{int}} = \tfrac{1}{2}\,\mathbb{E}\big[(1 - s_{\text{real}})_+ + (1 + s_{\text{fake}})_+\big], \qquad \mathcal{L}_G^{\text{int}} = \mathbb{E}[s_{\text{real}}] - \mathbb{E}[s_{\text{fake}}]. \end{gathered} \tag{7}$$

Each mixture is thus scored four times, and the two real and the two fake passes are averaged before the hinge. The reconstruction discriminator is an unconditional head that compares real images with their reconstructions, since a head conditioned on the source would classify any departure from it as fake, and conditioning is known to cause discriminator collapse on limited data (Shahbazi et al., 2022). It is separate from the interpolation discriminator, since mixtures are easier to detect as fake than reconstructions and could dominate a shared head. Table 26 gives the parameter counts and the schedule.

E.4 Decoder Architecture

MAETok's decoder has 12 ViT blocks of width 768 over 256 image tokens. Each of the six appended blocks computes $x + g\,(\mathrm{block}(x) - x)$ with a scalar gate g initialized at zero, and its rotary frequencies are copied from the pretrained decoder. The appended blocks run under activation checkpointing and add 43.1M parameters to the 173.9M counted in the released checkpoint, for which the authors publish 176M.

E.5 Pair Sampling

In each epoch, every training image is paired with its most similar unpaired image among its k nearest neighbours, measured by the cosine similarity of the CLS features of the frozen vision model. The number k grows from 8 to 128 over the epochs, Table 25, so the pairs become less similar as training proceeds. Only the 64 nearest neighbours are precomputed, and for $k > 64$ an image takes such a neighbour with probability $64/k$ and a random unpaired image otherwise. Each micro-batch holds four pairs, eight images, and mixes all 28 pairs of its images. Four of these mixtures therefore combine two paired, similar images, and the other 24 combine images from different pairs.

E.6 Interpolation Coefficients

Each of the 28 mixtures draws one $\alpha \sim \mathrm{Beta}(2, 2)$, with density $6\alpha(1 - \alpha)$, which favours the middle of the line, where decoding is hardest, Fig. 3a. We draw it by the exact inverse CDF

$$\alpha = \tfrac{1}{2} + \cos\!\left(\tfrac{\arccos(1-2u)+4\pi}{3}\right), \qquad u \sim \mathcal{U}[0, 1), \tag{8}$$

the trigonometric root of $3\alpha^2 - 2\alpha^3 = u$ on $[0, 1]$. The draw is reproducible, and no fixed values of α are inserted.

Table 27: **Evaluation datasets and protocols.** One row per measurement, with its corpus, sample count, pairing and reference.

Split	Units	Source and pairing	Evaluation target
ImageNet validation	50,000 midpoints	val code to a sampled top-10 nearest neighbour in the train split, slerp, $\alpha = 0.5$, the released evaluation code at its defaults, FID against the 50,000 validation originals	iFID, rFID, PSNR, SSIM, LPIPS
ImageNet validation	50,000 images	reconstruction, no pairing, the released evaluation code at its defaults for rFID and our own pixel evaluation, SSIM with a uniform window at data range 2, for the rest	rFID (Tables 1 and 11), PSNR, SSIM, LPIPS (Table 11)
ImageNet validation	50,000 reference images	equal-weight barycentres of $k = 2, 3, 5, 10$ codes, same-class and cross-class, straight line	FID, sFID, precision and recall of decoded barycentres (Table 12)
ImageNet validation	50,000 midpoints	random cross-class pairs under one seed, straight-line midpoint at $\alpha = 0.5$, ADM evaluator against the 50,000 validation originals	FID of decoded midpoints through the fine-tune (Table 8)
ImageNet validation	5,000 near and 5,000 random pairs	near pairs at DINOv3 cosine 0.6 to 0.8 and random pairs of different classes under one seed, the straight line decoded at nine equally spaced positions, LPIPS with AlexNet between consecutive decodes and to both source reconstructions	perceptual path length, largest-step share, midpoint distance and ordered fraction (Table 13)
MS-COCO train2017	50,000 images	reconstruction, no pairing	rFID, PSNR, SSIM, LPIPS (Table 11)
MS-COCO train2017	50,000 midpoints	one cross-pair derangement shared by every model, straight-line midpoint at $\alpha = 0.5$	FID of decoded midpoints, and the same ranking in four feature spaces (App. B.5)
MS-COCO val2017	5,000 images	reconstruction, no pairing, the split MAETok reports on	rFID against MAETok's published COCO value
ImageNet class-conditional	50,000 samples	SiT-B/1D, 400k steps, 250 SDE steps, no guidance	gFID (App. B.1)
ImageNet validation	50,000 images	MAE linear probe, 90 epochs, LARS	top-1 accuracy (Table 8)

E.7 Evaluation Protocol

Evaluators and samples. Every iFID and ImageNet rFID we measure in Tables 1 and 11 uses the released evaluation code of Xu et al. (2026) at its defaults, and daggered values are quoted, as each caption states. Table 27 gives the corpus, count, pairing and reference of every measurement, and Table 29 the evaluators.

Reproduction of released values. Table 28 reproduces the released iFID of MAETok and DM-VAE and the released rFID to within a few percent. These are reproduction gaps, not variance intervals. Chen et al. (2025a) report an rFID of 0.48 for these weights, under a protocol that names neither its reference statistics nor its toolkit.

Quoted comparisons. Table 29 names the quoted tokenizers, what supervised each, its corpus and its code dimensions, and every comparison holds the corpus and the evaluator fixed within a column. The daggered iFID values of Table 1 come from the comparison of Xu et al. (2026), which lists thirteen tokenizers in fourteen rows, since VA-VAE appears at two code widths. Table 1 omits the 64-channel VA-VAE (Yao et al., 2025) and the reconstruction-only QW-VAE, SD3-VAE, IN-VAE and FLUX-VAE, whose values Table 29 lists.

Pixel evaluation. Table 11 uses our own pixel evaluation on the same ImageNet images, since the released code maps images to $[0, 1]$ before treating them as $[-1, 1]$. It follows MAETok's SSIM convention, a uniform window at data range 2, and computes LPIPS on AlexNet. Table 30 gives the other conventions.

Pairing. The released iFID evaluation pairs each validation code with one of its ten nearest training codes, chosen with one seed for every model. It decodes the midpoint of each pair along a spherical path and computes the FID of the decoded midpoints. Training mixes codes along the straight line of Eq. 1. Our own comparison fixes the corpus, the reference, the ADM evaluator (Dhariwal & Nichol, 2021) and the straight line and varies only the pairing, Table 30. The frozen encoders that Table 1 sets apart are DINOv2 models pretrained on a dataset far larger than ImageNet. Neither MAETok nor InterpTok was trained on MS-COCO. Class readout adds 20-nearest-neighbour classification to the linear probe of Table 27.

Generator training and evaluation. Table 26 gives the SiT-B settings, and MAETok's authors report no SiT-B. Two draws of one generator differ by about 0.1 gFID, and a dash marks a value we do not use.

Table 28: **Reproducing published measurements.** Each row re-runs one published measurement under the protocol that produced it. †The published value.

Quantity	reported	ours	rel. diff.	source
iFID, MAETok	14.06†	14.42	+2.6%	released README of Xu et al. (2026)
iFID, DM-VAE	8.14†	8.29	+1.8%	released README of Xu et al. (2026)
rFID, MAETok, iFID protocol	0.58†	0.5819	+0.3%	released leaderboard of Xu et al. (2026)
rFID, MAETok, MS-COCO	4.87†	5.05	+3.8%	Table 4 of Chen et al. (2025a)

Table 29: **Evaluators and provenance of the quoted tokenizers.** The evaluator of every measurement, and for each quoted tokenizer what supervised it, on which corpus and at how many code dimensions.

Item	**Value**
Evaluators	
iFID and rFID	the released evaluation code of Xu et al. (2026) at its defaults
Inception features	the Inception weights of pytorch-fid and of torch-fidelity, which agree within 0.2 percent on identical decoded images at $n = 50{,}000$
Other FID comparisons	the ADM evaluator, against the validation originals for reconstruction and interpolation and the ADM ImageNet 256 reference statistics for generation
Pixel metrics	our own pixel evaluation, SSIM with a uniform window at data range 2, LPIPS with AlexNet
Encoder supervision, corpus and code dimensions	
RAE	frozen DINOv2 encoder with a trained decoder, ImageNet, 196,608 dimensions
DM-VAE	frozen DINOv2 encoder with a trained projector and decoder under a distribution-matching objective, ImageNet
MAETok	encoder trained on ImageNet against frozen DINOv2 and SigLIP features, 4,096 dimensions
VA-VAE, SoftVQ-VAE	encoder trained on ImageNet against frozen DINOv2 features, with MAE and CLIP reported as alternatives
l-DeTok	no foundation model, ImageNet
SD-VAE, EQ-VAE	no foundation model, OpenImages, EQ-VAE an equivariant fine-tune of SD-VAE. Table 1 quotes SD-VAE iFID for the Stable Diffusion 1.5 autoencoder, while our own SD-VAE rows use its released EMA fine-tune
Rows of the released comparison omitted from Table 1, with their released iFID	
QW-VAE	30.58
SD3-VAE	37.13
VA-VAE with 64 channels	37.14
IN-VAE	41.06
FLUX-VAE	67.41
REPA-E	SD-VAE tuned end to end with a diffusion transformer under representation alignment, ImageNet, from SD-VAE's OpenImages weights
InterpTok	MAETok's DINOv2-L and SigLIP supervision, the frozen DINOv3-B/16 vision model, a frozen DINO ViT-S/16 discriminator backbone and a representation Fréchet loss across three feature spaces, ImageNet, 4,096 dimensions

Table 30: **Other pixel conventions and other pairings.** ImageNet validation at $n = 50{,}000$ and InterpTok (82k) unless a row states otherwise.

Other pixel conventions	MAETok	InterpTok	SD-VAE
SSIM, Gaussian window at data range 1, ImageNet	0.644	0.670	0.708
SSIM, Gaussian window at data range 1, MS-COCO train2017	0.656	0.682	0.724

Measurement	MAETok	InterpTok
LPIPS with VGG, ImageNet	0.191	0.172
MS-SSIM, ImageNet, InterpTok (41k)	0.929	0.934
FID of decoded midpoints under other pairings		
each model's nearest validation neighbour	17.28	5.82
index-adjacent pairing under the dataset's class order	35.12	4.34
random cross-class barycentre sampler	87.49	14.82
the random-pair sampler that logged the fine-tune	87.34	14.92