APEX: ASYNCHRONOUS PREFIX DENOISING FOR EXTREME COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Fewer latent tokens promise cheaper diffusion. Yet aggressive spatial compression creates a reconstruction-generation dilemma: adding channels improves reconstruction but makes generation harder. Our goal is to make extreme spatial compression competitive by shifting this frontier. We introduce APEX, short for Asynchronous Prefix denoising for EXtreme compression, which pairs a hierarchical latent space with asynchronous denoising over channel groups. Our autoencoder orders channel prefixes from structure to detail, while one shared denoiser generates them at staggered local times without adding tokens. We show that naively combining a channel hierarchy with asynchronous denoising can make generation worse. APEX succeeds by matching what each level represents, when it denoises, and how training is allocated across the trajectory. On ImageNet-512, APEX outperforms existing methods at every evaluated channel count under $64\times$ and $128\times$ spatial compression, substantially improving this frontier. At matched training and guided sampling compute, it also outperforms a 256-token model at every evaluated budget, with larger gains at lower compute. APEX makes extreme spatial compression practical for compute-efficient latent diffusion.

1 INTRODUCTION

Modern diffusion models increasingly use Transformers, whose attention cost grows quadratically with the number of input tokens. Latent diffusion reduces this cost by compressing an image into a smaller spatial grid and applying diffusion there (Rombach et al., 2022). Our goal is to make extreme spatial compression competitive for latent diffusion. Stronger spatial compression produces fewer tokens. Channel count is a separate design choice, but preserving reconstruction quality typically requires adding channels as the grid shrinks, producing increasingly *wide* latents. This trade is attractive because shorter sequences can substantially reduce denoiser cost, permitting more optimization and sampling under a fixed compute budget. This suggests an obvious strategy: compress the spatial grid as aggressively as possible, then add enough channels to preserve reconstruction quality.

Unfortunately, this strategy encounters a well-established reconstruction-generation dilemma (Yao et al., 2025; Chen et al., 2025c;d; Lai et al., 2025). At fixed spatial compression, adding channels improves reconstruction but makes the resulting latent harder to generate. Additional channels let the encoder preserve more image-specific information, including exact textures and local details. This lowers reconstruction error, but increases the information that the diffusion model must produce (Chen et al., 2025d; Heek et al., 2026). Standard latent diffusion therefore faces a poor choice: use fewer channels and sacrifice reconstruction quality, or preserve more detail and sacrifice generation quality. Existing approaches reduce this tension by restricting latent information or importing pretrained semantic structure (Kingma & Welling, 2013; Yu et al., 2025; Yao et al., 2025; Zheng et al., 2025; Pan et al., 2025). The closest structured-latent method in our setting organizes channel prefixes within one reconstruction latent, but still denoises each retained prefix at a shared timestep (Chen et al., 2025d). This limitation becomes especially severe at $128\times$ spatial compression, where a 512^2 image is represented by a 4×4 grid of only 16 tokens.

APEX addresses this problem without imposing an additional bottleneck on the final latent. Instead, it changes how the channels are organized and generated. Coarse structure must be generated from scratch, while exact texture and local detail become easier to predict once the scene is established.

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.

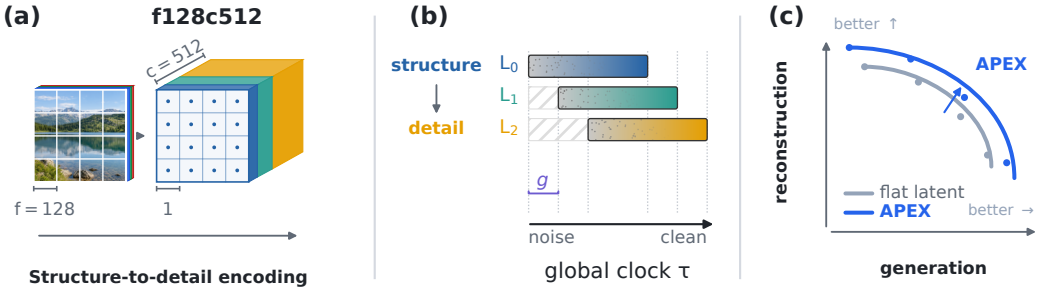


Figure 1: **APEX enables extreme-compression latent diffusion.** **(a)** An f128c512 latent represents a 512^2 image with 16 spatial tokens and orders channels from structure to detail. **(b)** One global clock denoises those levels at staggered local times; g is the delay before each later level starts. **(c)** Ordering generation this way improves the reconstruction–generation trade-off relative to a flat latent.

We therefore factorize generation into structure generation followed by conditional refinement. Our autoencoder orders channel prefixes from structure to detail by supervising them at progressively higher resolutions. During diffusion, one shared Transformer denoises the newly added channel groups at staggered local times, so each refinement is predicted from cleaner preceding context. As this is done along the channel dimension, APEX does not increase the number of tokens. Figure 1 summarizes the representation and denoising process. Although hierarchical channel latents and asynchronous denoising each exist, combining them is not automatic: shared-time diffusion performs poorly on APEX levels, while staggered local times perform poorly with the DC-AE 1.5 hierarchy and prefix sampler (Table 2).

We evaluate APEX on ImageNet-512 at $64\times$ and $128\times$ spatial compression. We write $f64c128$ for $64\times$ spatial compression with 128 channels. At 512^2, $f64$ and $f128$ correspond to 64 and 16 tokens. Across every evaluated channel count, APEX outperforms all compared methods and degrades far less as channels are added. At $f128c512$, APEX reaches 8.98 guided gFID from 16 tokens with SiT-XL, while every compared method remains in the double digits. Our 64-token $f64c128$ model matches the gFID of the widely used 1024-token $f8c4$ SD-VAE (Rombach et al., 2022) using 15% of its denoiser-training FLOPs (Appendix B.4). With training compute matched, our 64-token hierarchical model also outperforms a less-compressed 256-token model across every evaluated guided inference budget, with larger gains as sampling compute decreases. These results show that once the reconstruction-generation dilemma is addressed, extreme spatial compression becomes a quality-compute advantage rather than only a per-step speedup.

Our contributions are threefold:

1. **Making hierarchy and asynchronous denoising work together.** We show that nested channel levels and staggered local times do not combine automatically. APEX couples resolution-supervised levels, local times, and training allocation within one fixed-token latent.
2. **Turning high compression into a compute advantage.** With training compute matched, APEX at 64 tokens outperforms a less-compressed 256-token baseline across every evaluated guided inference budget, with larger gains at lower compute.
3. **Generating without an additional information bottleneck.** Rather than improving generative tractability by restricting what the latent can store, APEX improves generative tractability while maintaining comparable reconstruction quality, without imposing an additional explicit information bottleneck.

2 Related Work

Deep Compression Autoencoders. Deep-compression autoencoders reduce diffusion sequence length by moving representational capacity from spatial positions into latent channels. DC-AE made reconstruction practical at $32\times$, $64\times$, and $128\times$ spatial compression and provides our base

architecture (Chen et al., 2025c). DC-AE 1.5 structures channel prefixes and trains diffusion on randomly sampled prefixes at a shared timestep, while FreqWarm modifies diffusion training through a frequency curriculum (Chen et al., 2025d; Lai et al., 2025). These are our closest direct baselines. APEX instead pairs resolution-matched cumulative levels with staggered local times; APEX levels under shared-time diffusion and a direct application of the staggered schedule to the DC-AE 1.5 construction perform poorly (Table 2).

Latent Information and Semantic Alignment. Existing methods improve diffusibility either by controlling latent information or importing pretrained semantic structure. Traditional VAEs impose a Gaussian prior, Unified Latents explicitly controls information rate with a learned diffusion prior, and MAETok uses a non-variational discriminative latent for 128-token generation (Kingma & Welling, 2013; Heek et al., 2026; Chen et al., 2025b). Semantic-alignment methods instead shape the latent or denoiser using pretrained representations (Yu et al., 2025; Yao et al., 2025; Zheng et al., 2025; Kouzelis et al., 2025b; Wu et al., 2025; Leng et al., 2025). SFD and Learning When to Denoise additionally pair a DINOv2-derived semantic latent with a separate texture latent and denoise semantics first using a fixed or learned relative schedule (Pan et al., 2025; Qian & Cheng, 2026; Oquab et al., 2024). APEX uses no external semantic representation or explicit latent prior. It induces structure within one reconstruction latent, leaving later levels free to preserve image-specific detail.

Hierarchical and Frequency-Based Representations. Coarse-to-fine representations appear in Matryoshka learning, cascaded and nested diffusion, multiscale tokenizers, ordered-channel models, and flexible-length token sequences (Kusupati et al., 2022; Ho et al., 2022; Gu et al., 2024; Chen et al., 2025a; Zhong et al., 2025; Paul et al., 2026; Cai et al., 2026; Bachmann et al., 2025). Frequency-based methods instead reshape latent spectra or explicitly separate coarse and fine components (Ning et al., 2026; Skorokhodov et al., 2025; Kouzelis et al., 2025a; Li et al., 2025; Guan et al., 2026). Denoising-VAE constructs progressively low-pass latent levels with blurred image targets and frequency-aligned diffusion (Xiang et al., 2025); we use Spectrum Matching as the representative autoencoder-side baseline for this family. WaiT similarly delays wavelet-detail generation, but operates in pixel space rather than a learned high-compression latent (Pavasovic et al., 2026). APEX instead uses cumulative channel prefixes of one jointly decoded latent on a fixed spatial grid, supervising each prefix at a corresponding resolution and generating later groups from cleaner preceding levels.

3 METHOD

APEX has two components. First, a single autoencoder orders cumulative channel levels from structure to detail using progressively higher-resolution targets (Section 3.1). Second, a shared Transformer denoises the channels introduced at each level at staggered local times, allowing later levels to refine cleaner earlier ones without adding spatial tokens (Section 3.2; Figure 2). Pseudocode for autoencoder training, diffusion training, and inference is given in Appendices C and D (Algorithms 1, 2, and 3).

3.1 RESOLUTION-AWARE AUTOENCODER TRAINING

Given an image $x \in \mathbb{R}^{R\times R\times 3}$, the encoder produces a latent $z \in \mathbb{R}^{(R/f)\times(R/f)\times C}$, which we partition as

$$z = [b_0 \| b_1 \| \cdots \| b_{L-1}], \qquad b_\ell \in \mathbb{R}^{(R/f)\times(R/f)\times d_\ell}. \tag{1}$$

We call the cumulative representation a level:

$$z^{(\ell)} = [b_0 \| \cdots \| b_\ell] \in \mathbb{R}^{(R/f)\times(R/f)\times C_\ell}, \qquad C_\ell = \sum_{k=0}^{\ell} d_k, \tag{2}$$

It retains all preceding channels while adding capacity on the same spatial grid.

We use a simple structure-to-detail heuristic to choose target resolutions. At lower channel counts, a roughly $4\times$ increase in cumulative channels corresponds to a $2\times$ increase in target resolution along each spatial dimension. At larger widths, we treat this proportionality as a guideline rather than a

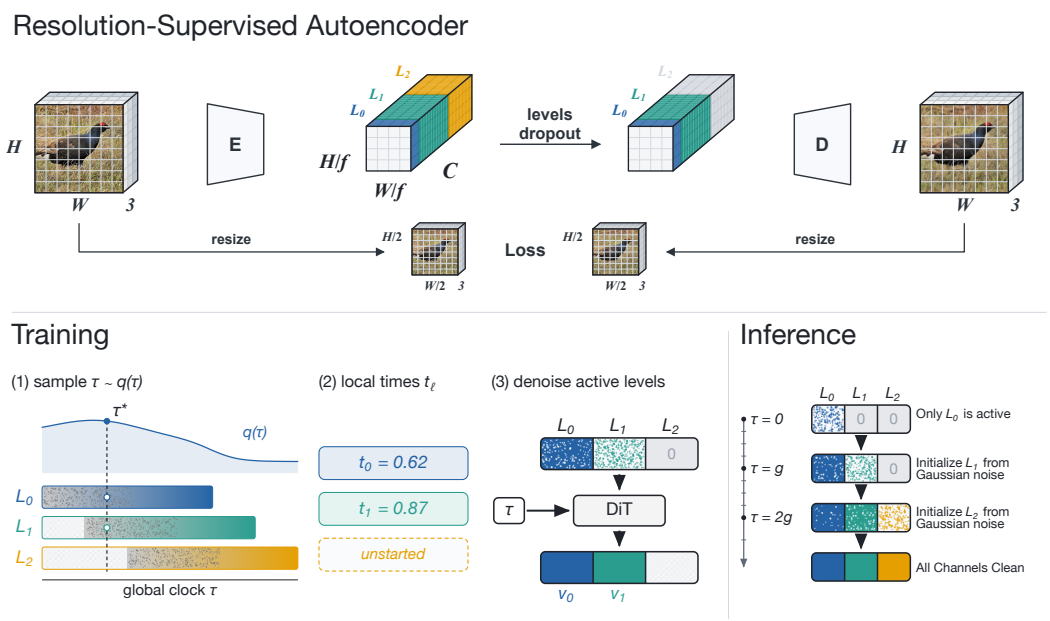


Figure 2: **Channel hierarchy.** We divide the channels of each latent token into ordered groups. The cumulative prefix through group ℓ defines level L_ℓ. **Resolution-supervised autoencoder.** We sample a level, zero all later groups, and decode the retained prefix at full resolution. Reconstruction and target are then resized to that level's assigned resolution before computing the autoencoder loss. **Diffusion training.** A global clock τ is sampled from $q(\tau)$ and mapped to one local timestep per group. Unstarted groups are zeroed, and the shared DiT predicts velocities for the denoising groups. **Inference.** Groups start in sequence along the same clock. Each newly active group is initialized from Gaussian noise, and generation continues until the full latent is clean.

constraint: we introduce intermediate channel levels to avoid excessively large refinement groups and use monotonically increasing target resolutions up to the full image resolution. Exact assignments are given in Appendix Table 5. Any additional levels are also supervised at full resolution. We denote a hierarchy by its spatial compression followed by its cumulative level sizes; for example, f64c8−32−128 has $64\times$ spatial compression and levels with 8, 32, and 128 cumulative channels, assigned to 128^2, 256^2, and 512^2 targets, respectively. At each training step, we sample a level uniformly,

$$\ell \sim \mathrm{Unif}\{0, \ldots, L-1\},$$

retain its cumulative representation $z^{(\ell)}$, and set all later channel blocks to zero. The shared decoder produces

$$\hat{x}^{(\ell)} = D([b_0 \| \cdots \| b_\ell \| 0 \| \cdots \| 0]) \,. \tag{3}$$

Although the decoder always outputs the original image resolution, we resize both the reconstruction and target to $R_\ell \times R_\ell$ before computing the loss:

$$\mathcal{L}_{\mathrm{AE}} = \mathbb{E}_{x,\ell}\left[\mathcal{L}_{\mathrm{base}}\left(P_\ell(\hat{x}^{(\ell)}), P_\ell(x)\right)\right], \tag{4}$$

where P_ℓ is an antialiased resize and $\mathcal{L}_{\mathrm{base}}$ follows the reconstruction, perceptual, and adversarial objective of the underlying DC-AE (Chen et al., 2025c). Exact channel counts, target resolutions, and training details are given in Section 4 and Appendix C.

Resolution-aware supervision discourages narrow levels from spending their limited capacity on isolated high-frequency features. When every level is supervised at full resolution, early reconstructions often preserve sharp edges without recovering the surrounding structure. Resizing the loss instead encourages early levels to preserve coherent layout and shape, while later levels restore the missing detail (Figure 7).

3.2 Staggered Channel Diffusion

During diffusion, each block receives its own local timestep while remaining on the same spatial grid.

A global clock and level states. A stagger schedule coordinates the levels through a single global clock τ: each level carries a local time $t_\ell(\tau)$ that decreases monotonically from 1 (noise) to 0 (clean) over its own window of the clock, and the windows are ordered so that

$$t_0(\tau) \;\le\; t_1(\tau) \;\le\; \cdots \;\le\; t_{L-1}(\tau). \qquad (5)$$

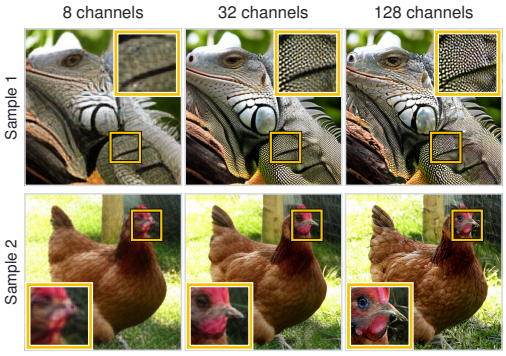


Figure 3: **Staged generation.** Decodes after each level reaches $t = 0$. Additional samples in Appendix A.1.

Earlier levels are therefore never noisier than the refinement levels they condition. At any clock value, each level is in one of three states: *unstarted* (its window has not begun), *denoising* ($0 < t_\ell(\tau) < 1$), or *done* ($t_\ell(\tau) = 0$). Unstarted blocks are zeroed in the Transformer input; denoising blocks are noised at their local times; done blocks appear clean and act as fixed context. The shared network predicts velocities for all blocks, but only denoising levels are supervised and updated. Predictions on unstarted and done blocks are discarded.

Equal-gap stagger. Our main schedule delays consecutive levels by a constant gap g:

$$t_\ell(\tau) = \mathrm{clip}\big(1 + \ell \cdot g - \tau,\, 0,\, 1\big), \qquad \tau \in [0,\, 1 + (L-1)g], \qquad (6)$$

so level ℓ is unstarted for $\tau \le \ell \cdot g$ (a multiple of the gap), denoising for $\ell \cdot g < \tau < 1 + \ell \cdot g$, and done afterward: the first level begins immediately, each later level starts after an additional delay g, and earlier levels are always further along their trajectories, providing cleaner context for refinement. The formulation is schedule-agnostic, and any monotone local clocks satisfying Eq. 5 are valid. We adopt the equal-gap form throughout for its simplicity.

Training. We draw the clock from a sampling density $\tau \sim q$ and noise every denoising block with the standard flow-matching interpolation (Lipman et al., 2023; Liu et al., 2022), each at its own local time:

$$\tilde{b}_\ell = (1 - t_\ell)\, b_\ell + t_\ell\, \epsilon_\ell, \qquad v_\ell = \epsilon_\ell - b_\ell, \qquad \epsilon_\ell \sim \mathcal{N}(0, I). \qquad (7)$$

The objective weights the per-level velocity errors,

$$\mathcal{L}(\tau) = \sum_\ell w_\ell(\tau)\, \big\|\hat{v}_\ell - v_\ell\big\|^2, \qquad (8)$$

where the error of each level is averaged over its spatial positions and channels. We set $w_\ell(\tau)$ to zero outside the denoising state and constant across denoising levels (normalized to sum to one), which prevents wide refinement levels from dominating training solely because they contain more channels.

The schedule fixes the levels' relative progress; q decides how often each part of the trajectory is trained (Figure 4). Because L_0 must learn structure from scratch, we assign probability p_{L_0} to $0 < \tau < 1$, where L_0 is denoising, and spread the remaining probability uniformly over the rest of the clock. A sample from this interval also trains every later level whose window has already opened. Section 4.4 analyzes the effect of this sampling mechanism.

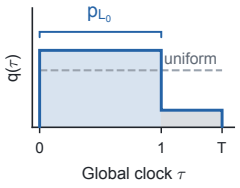


Figure 4: Global clock density.

Sampling. Inference discretizes the clock into a grid $0 = \tau_0 < \cdots < \tau_N$ that includes each activation point $\ell \cdot g$ (multiples of the gap).

At every step, one shared network evaluation serves all levels, and each block advances by its own time increment:

$$\tilde{b}_\ell \;\leftarrow\; \tilde{b}_\ell + \big(t_\ell(\tau_{k+1}) - t_\ell(\tau_k)\big)\, \hat{v}_\ell. \qquad (9)$$

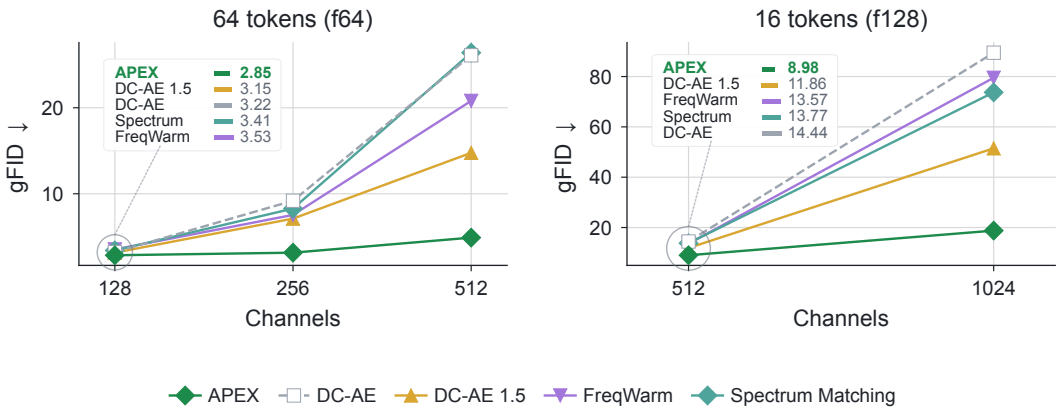


Figure 5: **APEX degrades far more slowly as latent width increases.** Left: 64-token f64 latents; right: 16-token f128 latents. At f64, APEX changes from $2.85 \rightarrow 3.15 \rightarrow 4.90$ gFID as channels increase $128 \rightarrow 256 \rightarrow 512$, while DC-AE degrades from $3.22 \rightarrow 9.19 \rightarrow 26.10$. At f128, increasing channels from $512 \rightarrow 1024$ changes APEX from $8.98 \rightarrow 18.77$, compared with DC-AE $14.44 \rightarrow 89.47$ and DC-AE 1.5 $11.86 \rightarrow 51.47$. The boxed values summarize performance at f64c128 and f128c512. All runs use matched denoiser-training FLOPs and total NFE.

Unstarted and done levels have constant local time, so their increments vanish and they do not move. When a level's window opens, its block is initialized from Gaussian noise (Algorithm 3, Appendix D). Under the equal-gap schedule the increment is $-\Delta\tau$ for every denoising level and zero otherwise, so each step reduces to a single shared Euler update over the denoising blocks: one network evaluation per step, the same cost per step as a flat model. A nonlinear clock changes only the increments in Eq. 9. When each level reaches $t = 0$, we decode the cumulative prefix. Later levels add detail that is absent from the 8-channel prefix (Figure 3; Appendix A.1). Successive-prefix differences are shown in Appendix A.2.

DiT initialization. We initialize the rows of the input projection associated with b_0 using the standard SiT initialization (Ma et al., 2024) and the rows associated with $b_1, \ldots, b_{L-1}$ to zero. The model therefore begins as an L_0-only generator and gradually learns to incorporate refinement information, without modifying the Transformer trunk.

4 Experiments

We target the operating point where each token must absorb substantial instance-specific detail: high native resolution under aggressive spatial compression. Following DC-AE (Chen et al., 2025c), we evaluate on ImageNet-512 (Deng et al., 2009) at $64\times$ and $128\times$ spatial compression. We first compare reconstruction and generation across f64 and f128 latents with increasing channel capacity, then test whether high compression remains advantageous when both training and inference are compute-constrained. We next perform an extensive component study showing that the gain emerges from the complete representation–diffusion co-design: neither the hierarchical latent nor asynchronous denoising reaches the same quality on its own. Finally, we study the schedule choices used by the main models.

Setup. We denote an autoencoder with $f\times$ spatial compression and C channels by $\mathrm{f}f\mathrm{c}C$. At 512^2, f32, f64, and f128 produce 16×16, 8×8, and 4×4 latent grids, corresponding to 256, 64, and 16 Transformer tokens. APEX uses c8−32−128 at f64 and c32−128−256−512 at f128; wider variants append c256 and c512 at f64 and c1024 at f128. Appendix Table 5 gives the exact groups and target resolutions. Unless stated otherwise, we use SiT-XL/1 (Ma et al., 2024), with SiT-B/1 for component and design ablations. We report rFID on the ImageNet validation set and gFID from 50K class-conditional samples using the ADM protocol (Dhariwal & Nichol, 2021). Generation uses Euler sampling and CFG (Ho & Salimans, 2022) with scale 1.5 unless otherwise specified; Appendix E additionally reports unguided gFID, sFID, and Inception Score. Within each

comparison, we match cumulative denoiser-training FLOPs and, unless varied, total NFEs; methods sharing a latent shape also use the same denoiser, optimizer, batch, and evaluation protocol. Full configurations and compute accounting are in Appendix B.

Baselines. DC-AE uses the unstructured reconstruction latent of Chen et al. (2025c). DC-AE 1.5 trains cumulative channel prefixes and augments diffusion with randomly sampled prefixes under a shared timestep (Chen et al., 2025d). FreqWarm changes the frequency curriculum of the diffusion model without modifying the autoencoder (Lai et al., 2025). Spectrum Matching regularizes the latent and reconstruction spectra (Ning et al., 2026). We implement all baselines in the same DC-AE and SiT training framework.

4.1 Generation Across Channel Counts

Following DC-AE 1.5 (Chen et al., 2025d), we find that imposing a channel hierarchy has little effect on complete-latent reconstruction, so we match all methods at each latent geometry to the same rFID. We use the official DC-AE f64c128 and f128c512 models as reference points and train flat DC-AE models at the additional widths using the same architecture and recipe (Table 1). We evaluate c128, c256, and c512 at f64 and c512 and c1024 at f128, matching cumulative denoiser-training FLOPs and total NFE across methods.

Table 1: **Matched full-latent reconstruction quality.** Every method at a given latent geometry is trained to the same rFID shown below.

Latent	Matched rFID ↓	PSNR ↑	Reference
f64c128	0.22	26.15	Official DC-AE
f64c256	0.18	26.8	Trained by us
f64c512	0.13	27.30	Trained by us
f128c512	0.23	25.73	Official DC-AE
f128c1024	0.20	26.1	Trained by us

Figure 5 shows that APEX degrades far more slowly as the latent widens. At f64, APEX gFID changes only from 2.85 at c128 to 3.15 at c256 and 4.9 at c512, while DC-AE rises from 3.22 to 9.19 and 26.1. At f128, APEX reaches 8.98 and 18.77 at c512 and c1024, versus $14.44/89.47$ for DC-AE and $11.86/51.47$ for DC-AE 1.5. At f128c1024, Spectrum Matching and FreqWarm reach 73.71 and 79.51, respectively.

Unguided APEX results across channel counts are in Appendix E.1.

4.2 APEX Makes High Compression Advantageous Under Constrained Compute

APEX turns the lower per-step cost of extreme compression into a genuine quality advantage. We compare APEX f64c8−32−128 with DC-AE f64c128 and f32c32 baselines using the same denoiser architecture and matched cumulative denoiser-training FLOPs. The f64 models are trained for 500K steps with batch size 2048, while f32c32 is trained for 250K steps with batch size 1024. One f32 denoiser evaluation costs approximately $4\times$ one f64 evaluation, so a matched inference budget affords the f64 models approximately four times as many NFEs. For each model, we select the best evaluated CFG value independently at each inference budget. Figure 6(b) shows the full sweep at the $1\times$ operating point, where APEX and f32c32 select CFG 1.4, while flat f64c128 selects 1.5.

Table 2: **Building APEX from DC-AE.** The first block successively introduces APEX components; the second removes components from the complete method. All runs use f64c128 and SiT-B/1 at matched compute, and ΔgFID is measured relative to APEX.

Configuration	gFID ↓	ΔgFID
Build APEX from DC-AE		
DC-AE	13.27	+5.15
+ APEX levels	20.37	+12.25
+ cumulative-prefix diffusion training	17.83	+9.71
+ **APEX diffusion**	**8.12**	**0.00**
Remove components from APEX		
− matched-resolution targets	8.77	+0.65
− APEX levels and clock sampler *(DCAE1.5 + naive async schedule)*	52.89	+44.77

Figure 6(a) shows that compression alone does not guarantee a better quality–compute trade-off. The flat f64 baseline is preferable only at the smallest sampling budgets; by $1/2\times$ relative compute, f32c32 is already better despite receiving one quarter as many evaluations. APEX removes this crossover: it outperforms f32c32 at every matched guided inference budget, with the margin increasing as compute becomes more constrained. At the $1\times$ budget, APEX reaches a gFID of 2.74

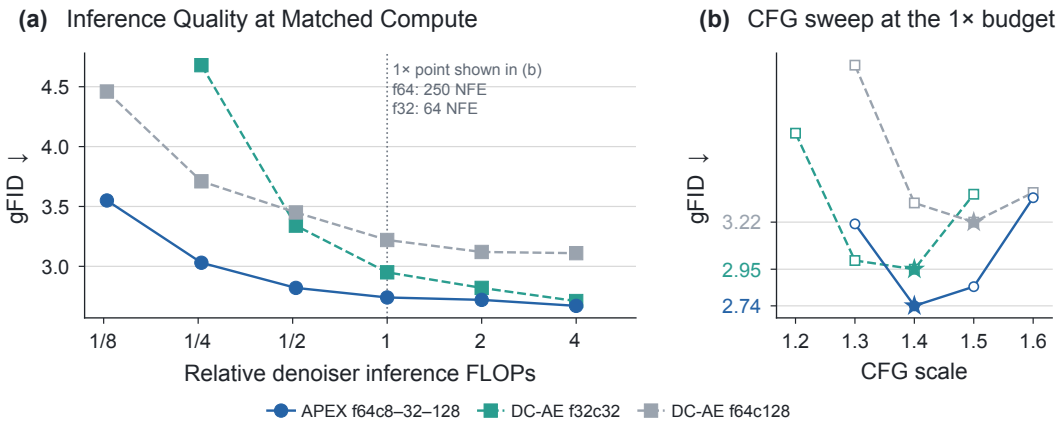


Figure 6: **APEX makes high compression advantageous under constrained compute. (a)** Guided gFID versus relative denoiser inference FLOPs, where $1\times$ matches f64 at 250 NFE with f32 at 64 NFE. **(b)** CFG sweep at the $1\times$ budget. Stars mark the selected guidance scale for each model.

with 250 NFEs, compared with 2.95 for f32c32 with 64 NFEs. At 1000 NFEs ($4\times$), APEX reaches 2.67. APEX therefore makes high compression advantageous across the full evaluated compute-constrained regime.

4.3 The Hierarchy and Denoising Process Must Work Together

The complete APEX design delivers a clear performance jump over every partial construction. We extensively ablate the representation, autoencoder targets, channel partition, diffusion schedule, loss aggregation, and training sampler at f64c128 using SiT-B/1. All variants use the same batch, evaluation setting, and cumulative denoiser-training FLOPs. We study representative controls obtained by using our hierarchy without asynchronous generation, and by applying asynchronous generation without the matching hierarchy or sampling regime.

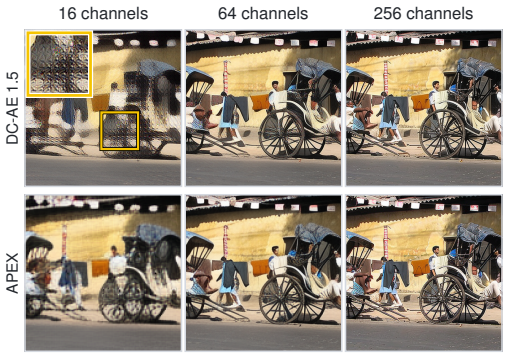


Figure 7: **Prefix reconstructions.** DC-AE 1.5 (top) vs APEX (bottom) prefixes. High-frequency detail leaks into DC-AE 1.5's early levels.

Here, APEX levels denote f64c8-32-128 trained with matched-resolution targets. APEX diffusion combines staggered local times, level-balanced losses, and the APEX clock sampler. Table 2 first builds APEX from DC-AE, then removes key components from the complete method. Appendix E.2 reports sFID and IS for the same variants.

The cumulative path shows that the hierarchy alone is insufficient. Adding APEX levels to ordinary diffusion worsens gFID, and cumulative-prefix training recovers only part of this loss. The large improvement appears when the matching diffusion process is introduced, reducing gFID from 17.83 to 8.12. The subtractive controls tell the same story. Removing matched-resolution targets raises gFID to 8.77, while naively replacing the APEX levels and clock sampler with the DC-AE 1.5 construction of many small uniform-width levels fails severely at 52.89. Figure 7 shows the corresponding representational difference: APEX prefixes are coherent and usable at lower resolutions, whereas DC-AE 1.5 prefixes mix coarse structure with isolated high-frequency detail.

4.4 Design Choices

Two choices are critical in practice: the stagger gap and the allocation of training probability along the global clock. Unless stated otherwise, these experiments use f64c128 and SiT-B/1. Only the

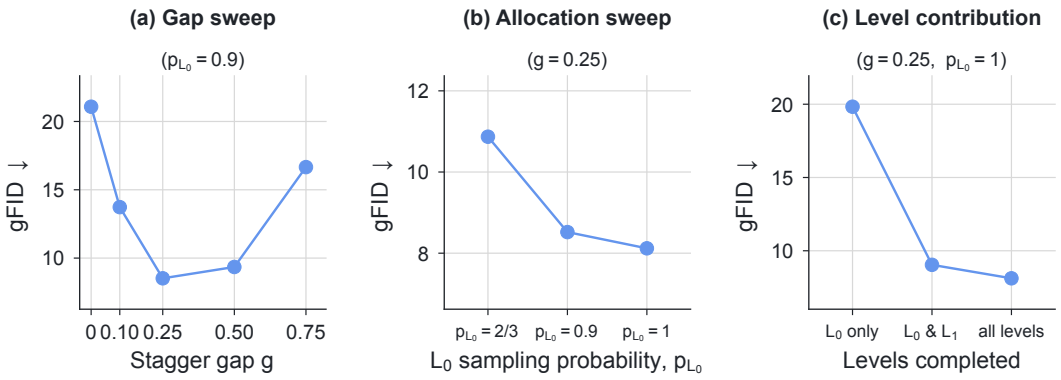


Figure 8: **Gap, allocation, and level contribution.** **(a)** We vary the stagger gap while using $p_{L_0} = 0.9$. At $g = 0$, the tail interval is absent and the process reduces to synchronous diffusion. **(b)** Holding $g = 0.25$, we vary p_{L_0}, the training probability assigned to the interval where L_0 is denoising. Uniform clock sampling corresponds to $p_{L_0} = 2/3$. **(c)** For the $g = 0.25$, $p_{L_0} = 1$ checkpoint, we stop inference when L_0, L_1, or L_2 completes and redistribute the same 250 NFEs over the retained clock interval. Generation improves at every endpoint, showing that all three levels contribute.

denoiser size and the explicitly ablated variable change. Appendix E.2 additionally reports unguided gFID for the gap and allocation sweeps, together with sFID and IS for all three analyses.

Stagger gap. Increasing g gives later levels cleaner structural context, but under our equal-gap schedule it also lengthens the global trajectory and reduces overlap, making a fixed NFE budget less efficient. Figure 8(a) shows that this trade-off is optimized near $g = 0.25$.

Training allocation. L_0 carries the unconditional generation burden, whereas later levels solve easier conditional refinements. We therefore place mass p_{L_0} on the interval where L_0 is active and distribute the remainder uniformly over the rest of the clock (Figure 4). Figure 8(b) shows that increasingly structure-heavy allocation improves gFID, with $p_{L_0} = 1$ performing best even though later portions of the clock receive no direct supervision. For the main models, we use $p_{L_0} = 0.9$ rather than 1. The $p_{L_0} = 1$ result above uses only three levels; in deeper hierarchies, later levels begin at or after $\tau = 1$ and would receive little or no direct supervision. Using 0.9 preserves tail coverage while still concentrating training on structure generation.

To quantify the contribution of each level, we evaluate the $p_{L_0} = 1$ model with the same 250-NFE budget redistributed over successively longer prefixes of its three-level clock. Stopping after the first one, two, or three levels reach $t = 0$, and decoding the corresponding cumulative latent, gives gFID 19.83, 9.04, and 8.12, respectively. These results confirm that the model uses the full hierarchy: later levels require far less direct supervision than L_0, yet remain essential to final generation quality. In particular, L_2 still contributes despite receiving no direct supervision over the latter half of its trajectory and despite the modest SiT-B/1 denoiser used here.

5 Conclusion and Limitations

We introduced APEX, which pairs a resolution-aware hierarchy of channel prefixes with asynchronous denoising across channels. At $64\times$ and $128\times$ spatial compression, APEX substantially shifts the reconstruction–generation frontier as channels are added. With matched training compute, it also outperforms a less-compressed 256-token baseline across all evaluated guided inference budgets, making extreme compression competitive under constrained compute.

APEX still requires choosing channel boundaries, per-level target resolutions, the stagger gap, and the global-clock distribution. Section 4.4 provides practical guidance, but the optimal choices may vary across settings. Learning them jointly with the autoencoder and denoiser is a promising direction.

AI USE STATEMENT

In this work, we used generative AI tools to help refine the research framing and hypotheses, design and critique experiments and ablations, interpret results, and draft portions of the manuscript. We also used generative AI tools to discover and summarize related work, improve the clarity of the writing, and design explanatory figures. We did not use generative AI tools to generate experimental data, reported numerical results, or mathematical proofs. The authors verified all suggested citations against the original sources, checked all empirical claims against the measured results, and reviewed and revised all AI-assisted text and figures. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The appendices provide complete autoencoder and diffusion training details, latent configurations, sampling and evaluation protocols, FLOP accounting, training and inference pseudocode, and figure-specific reproduction instructions. Appendix F records the protocol and data required to reproduce each main-text figure. We will release the training, evaluation, and figure-generation code, together with all configuration files, with the public version of the paper.

REFERENCES

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 *Proceedings of Machine Learning Research*, pp. 2241–2292. PMLR, 2025. URL https://proceedings.mlr.press/v267/bachmann25a.html.

Xin Cai, Zhiyuan You, Zhoutong Zhang, and Tianfan Xue. Da-vae: Plug-in latent compression for diffusion via detail alignment, 2026. URL https://arxiv.org/abs/2603.22125.

Cong Chen, Ziyuan Huang, Cheng Zou, Muzhi Zhu, Kaixiang Ji, Jiajia Liu, Jingdong Chen, Hao Chen, and Chunhua Shen. Hieratok: Multi-scale visual tokenizer improves image reconstruction and generation, 2025a. URL https://arxiv.org/abs/2509.23736.

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 *Proceedings of Machine Learning Research*, pp. 8145–8171. PMLR, 2025b. URL https://proceedings.mlr.press/v267/chen25v.html.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *International Conference on Learning Representations*, 2025c. URL https://arxiv.org/abs/2410.10733. ICLR 2025.

Junyu Chen, Dongyun Zou, Wenkun He, Junsong Chen, Enze Xie, Song Han, and Han Cai. Dc-ae 1.5: Accelerating diffusion model convergence with structured latent space. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1–10, 2025d. doi: 10.1109/iccv51701.2025.01825. URL https://arxiv.org/abs/2508.00413.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/cvpr.2009.5206848.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2105.05233.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Joshua Susskind, and Navdeep Jaitly. Matryoshka diffusion models. In *International Conference on Learning Representations*, pp. 30234–30262, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/81166fbd9cc5adf14031cdb69d3fd6a8-Paper-Conference.pdf. ICLR 2024.

Jiarui Guan, Wenshuai Zhao, Zhengtao Zou, Juho Kannala, and Arno Solin. Latent-compressed variational autoencoder for video diffusion models, 2026. URL https://arxiv.org/abs/2604.16479. Accepted to CVPR 2026 findings.

Jonathan Heek, Emiel Hoogeboom, Thomas Mensink, and Tim Salimans. Unified latents (ul): How to train your latents, 2026. URL https://arxiv.org/abs/2602.17270.

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 30 (NIPS 2017)*, 2017. URL https://arxiv.org/abs/1706.08500.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022. URL https://jmlr.org/papers/v23/21-0635.html.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. URL https://arxiv.org/abs/1312.6114.

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 *Proceedings of Machine Learning Research*, pp. 31648–31666. PMLR, 2025a. URL https://proceedings.mlr.press/v267/kouzelis25a.html.

Theodoros Kouzelis, Efstathios Karypidis, Ioannis Kakogeorgiou, Spyridon Gidaris, and Nikos Komodakis. Boosting generative image modeling via joint image-feature synthesis. In *Advances in Neural Information Processing Systems 38*, pp. 18836–18865, 2025b. doi: 10.52202/085713-0563. URL https://arxiv.org/abs/2504.16064.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems 35*, pp. 30233–30249, 2022. doi: 10.52202/068431-2192. URL https://arxiv.org/abs/2205.13147.

Bolin Lai, Xudong Wang, Saketh Rambhatla, James M. Rehg, Zsolt Kira, Rohit Girdhar, and Ishan Misra. Toward diffusible high-dimensional latent spaces: A frequency perspective, 2025. URL https://arxiv.org/abs/2511.22249.

Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. REPA-E: Unlocking VAE for end-to-end tuning with latent diffusion transformers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18262–18272, 2025. doi: 10.1109/iccv51701.2025.01697. URL https://arxiv.org/abs/2504.10483.

Zongjian Li, Bin Lin, Yang Ye, Liuhan Chen, Xinhua Cheng, Shenghai Yuan, and Li Yuan. Wf-vae: Enhancing video vae by wavelet-driven energy flow for latent video diffusion model. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17778–17788, 2025. doi: 10.1109/cvpr52734.2025.01656. URL https://arxiv.org/abs/2411.17459.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747. ICLR 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

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 *Computer Vision – ECCV 2024*, pp. 23–40, 2024. doi: 10.1007/978-3-031-72980-5_2. URL https://arxiv.org/abs/2401.08740.

Charlie Nash, Jacob Menick, Sander Dieleman, and Peter Battaglia. Generating images with sparse representations. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7958–7968. PMLR, 2021. URL https://proceedings.mlr.press/v139/nash21a.html.

Mang Ning, Mingxiao Li, Le Zhang, Lanmiao Liu, Matthew B. Blaschko, Albert Ali Salah, and Itir Onal Ertugrul. Spectrum matching: a unified perspective for superior diffusability in latent diffusion, 2026. URL https://arxiv.org/abs/2603.14645.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2304.07193.

Yueming Pan, Ruoyu Feng, Qi Dai, Yuqi Wang, Wenfeng Lin, Mingyu Guo, Chong Luo, and Nanning Zheng. Semantics lead the way: Harmonizing semantic and texture modeling with asynchronous latent diffusion, 2025. URL https://arxiv.org/abs/2512.04926.

Sukriti Paul, Arpit Bansal, and Tom Goldstein. Channeltok: Efficient flexible-length vision tokenization, 2026. URL https://arxiv.org/abs/2606.04461.

Krunoslav Lehman Pavasovic, Théophane Vallaeys, Stéphane Mallat, Giulio Biroli, Luke Zettlemoyer, Brian Karrer, and Jakob Verbeek. Wait for the signal: Simple frequency-aware flow-matching, 2026. URL https://arxiv.org/abs/2607.28760.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://arxiv.org/abs/2212.09748.

Bingshuo Qian and Xiang Cheng. Learning when to denoise: Optimizing asynchronous schedules for latent diffusion, 2026. URL https://arxiv.org/abs/2606.19662.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685, 2022. doi: 10.1109/cvpr52688.2022.01042. URL https://arxiv.org/abs/2112.10752.

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 29 (NIPS 2016)*, 2016. URL https://arxiv.org/abs/1606.03498.

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 *Proceedings of Machine Learning Research*, pp. 55876–55905. PMLR, 2025. URL https://proceedings.mlr.press/v267/skorokhodov25a.html.

Ge Wu, Shen Zhang, Ruijing Shi, Shanghua Gao, Zhenyuan Chen, Lei Wang, Zhaowei Chen, Hongcheng Gao, Yao Tang, jian Yang, Ming-Ming Cheng, and Xiang Li. Representation entanglement for generation: Training diffusion transformers is much easier than you think. In *Advances in Neural Information Processing Systems 38*, pp. 8622–8651, 2025. doi: 10.52202/085713-0264. URL https://arxiv.org/abs/2507.01467.

Xunzhi Xiang, Xingye Tian, Guiyu Zhang, Yabo Chen, Shaofeng Zhang, Xuebo Wang, Xin Tao, and Qi Fan. Denoising vision transformer autoencoder with spectral self-regularization, 2025. URL https://arxiv.org/abs/2511.12633.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15703–15712, 2025. doi: 10.1109/cvpr52734.2025.01464. URL https://arxiv.org/abs/2501.01423.

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. URL https://arxiv.org/abs/2410.06940. ICLR 2025 Oral.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. doi: 10.1109/CVPR.2018.00068. URL https://arxiv.org/abs/1801.03924.

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

Tianxiong Zhong, Xingye Tian, Xuebo Wang, Boyuan Jiang, Xin Tao, and Pengfei Wan. Decoupling complexity from scale in latent diffusion model, 2025. URL https://arxiv.org/abs/2511.16117.

A Additional Visualizations

A.1 Generated prefixes

Each row is one generated sample. Columns show the image decoded when that level first reaches $t = 0$.

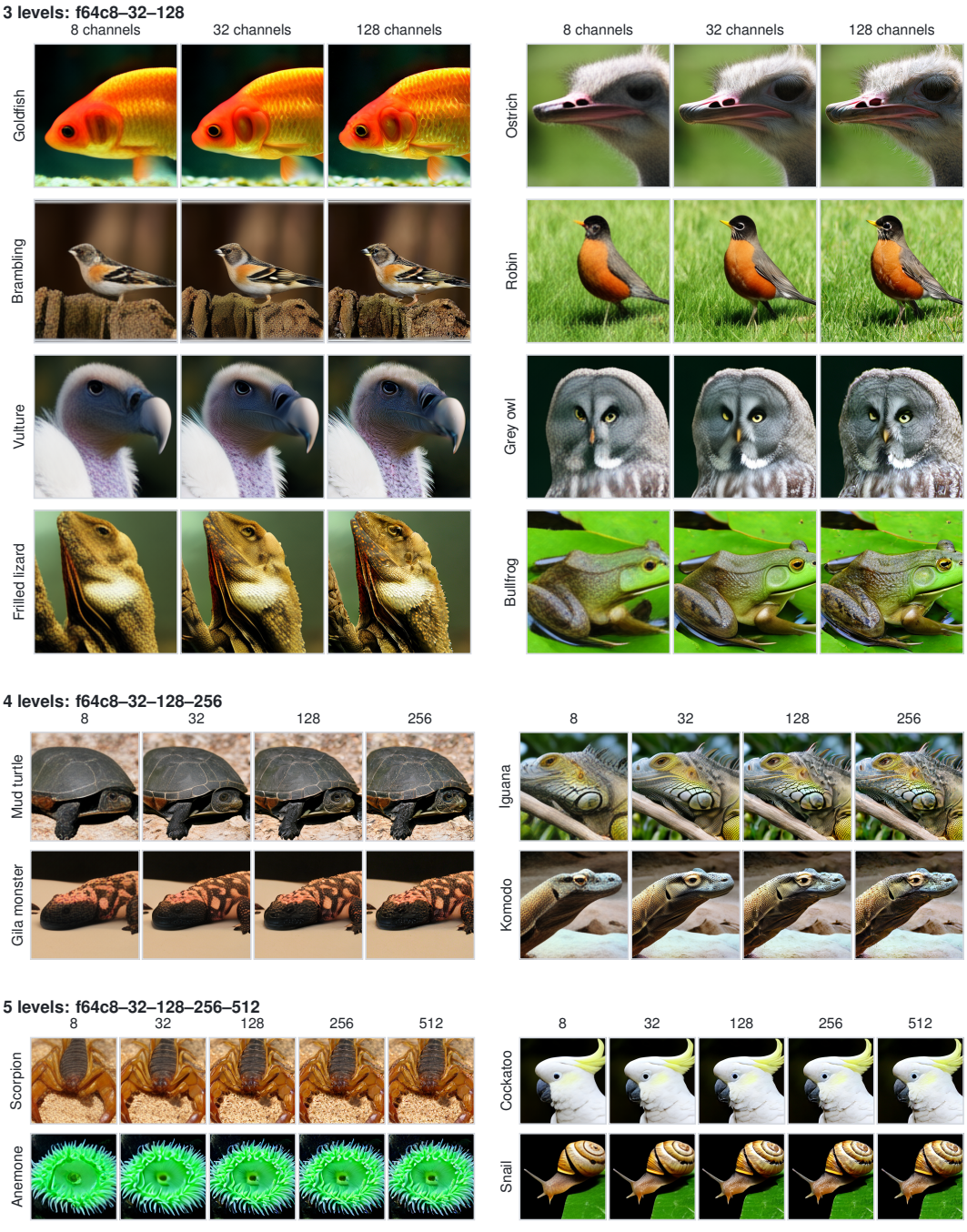
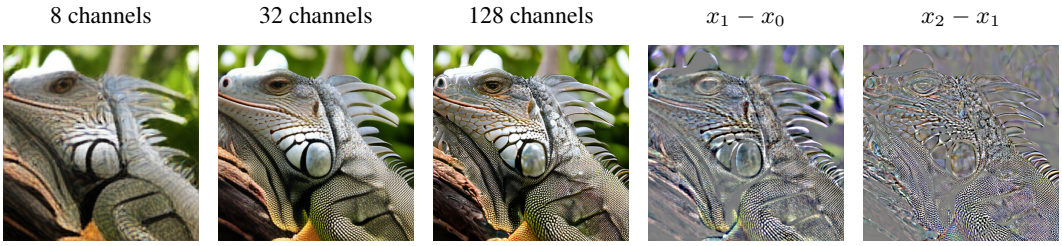


Figure 9: **Generated prefixes.** Each row is one sample from the staggered generator. We decode the completed prefix when each successive level reaches $t = 0$.

A.2 Refinement differences

For the two generated examples in Figure 3, we show successive prefix decodes and their differences. Here x_0, x_1, and x_2 are the images decoded from 8, 32, and 128 channels.

Sample 1



Sample 2



Figure 10: Successive generated prefixes and the changes introduced by later levels. Both samples use the same difference scale, with zero shown as gray.

Differences are computed per channel on the full-resolution decodes, and mapped to gray-centered images with a shared display scale of ± 0.5 in the $[0, 1]$ RGB range, which clips 1.3% of pixels.

B Extended Experimental Details

B.1 Compute-matched training

Within each experiment, we match *cumulative denoiser-training FLOPs* across all compared methods. This applies when changing the spatial compression, channel count, or denoiser size. We adjust the number of training steps for the model cost and global batch size; the comparison is not based on equal numbers of updates or images seen.

Table 3 gives the schedules for the main SiT-XL/1 comparisons. Figure 5 uses the f64 and f128 settings, and Figure 6 uses the f32 and f64 settings. All three schedules correspond to approximately 1.8×10^{20} denoiser-training FLOPs.

Table 3: Main diffusion-training schedules for ImageNet-512. Batch sizes are global.

Latent	Tokens	Batch size	Training steps	Training FLOPs
f32c32	256	1,024	250,000	$\approx 1.8 \times 10^{20}$
f64, all widths	64	2,048	500,000	$\approx 1.8 \times 10^{20}$
f128, all widths	16	4,096	1,000,000	$\approx 1.8 \times 10^{20}$

FLOP calculation. Following DiT (Peebles & Xie, 2023), we compute the training cost arithmetically as

$$F_{\text{train}} \simeq 3 F_{\text{forward}} BS, \tag{10}$$

where F_{forward} is the cost of one denoiser forward pass on one image, B is the global batch size, and S is the number of updates. A multiply-add counts as two FLOPs. The factor of three approximates the forward and backward passes. At these sequence lengths, the dominant cost is approximately linear in the number of tokens. Autoencoder training is separate from this budget.

B.2 MODELS AND OPTIMIZATION

We train on the ImageNet-1K training split at 512×512 (Deng et al., 2009). Main experiments use class-conditional SiT-XL/1; component and schedule ablations use SiT-B/1. Patch size 1 assigns one token to each latent spatial location. We follow the DC-AE diffusion-training setup and the underlying SiT optimization recipe (Chen et al., 2025c; Ma et al., 2024): AdamW (Loshchilov & Hutter, 2019) with a constant learning rate of 10^{-4} and an exponential moving average with decay 0.9999. Batch sizes and training lengths are given in Table 3.

Baselines. All methods use the same DC-AE and SiT training framework. DC-AE uses an unstructured latent (Chen et al., 2025c); DC-AE 1.5 uses cumulative prefixes and shared-timestep prefix training (Chen et al., 2025d). FreqWarm applies its frequency curriculum to diffusion training (Lai et al., 2025), and Spectrum Matching regularizes the latent and reconstruction spectra (Ning et al., 2026). Within a comparison, the backbone, optimizer, training compute, and evaluation protocol are shared.

B.3 SAMPLING AND EVALUATION

We evaluate generation using 50,000 class-conditional samples and the ADM protocol (Dhariwal & Nichol, 2021), reporting gFID (Heusel et al., 2017), sFID (Nash et al., 2021), and Inception Score (IS) (Salimans et al., 2016). We use classifier-free guidance with scale 1.5 unless another scale is specified (Ho & Salimans, 2022). Unguided sampling retains the class label but does not apply guidance (CFG 1.0). We report unguided gFID for the channel-scaling comparisons wherever the guided checkpoint is identified, and for the SiT-B gap and clock-allocation ablations, using the same trained model and number of sampling steps as the guided evaluation. Reconstruction rFID and PSNR are evaluated on the ImageNet validation split using the complete latent, following DC-AE (Chen et al., 2025c).

For N sampling steps, denoiser inference cost is NF_{forward} without guidance and $2NF_{\text{forward}}$ with guidance, excluding the autoencoder decoder. One f32 evaluation costs approximately four times an f64 evaluation. The $1\times$ budget in Figure 6 therefore pairs 64 f32 evaluations with 250 f64 evaluations. APEX sampling follows the Euler update in Section D; sampling budgets are given with the individual comparisons.

B.4 COMPARISON TO F8C4 SD-VAE

The standard 1024-token latent is the f8c4 SD-VAE used by SiT (Rombach et al., 2022; Ma et al., 2024). The published SiT-XL/2 ImageNet-512 result reaches gFID 2.62 after $3{,}000{,}000$ steps at batch size 256, using CFG 1.5 and 250 Euler–Maruyama steps (Ma et al., 2024). Under Equation 10 this is 1.21×10^{21} denoiser-training FLOPs.

We train SiT-XL/1 on APEX f64c8$-$32$-$128 with the optimizer in Section B. At $1{,}000{,}000$ steps and batch size $2{,}048$, this run reaches gFID 2.62 using 1.78×10^{20} FLOPs, or 15% of the f8c4 training compute. The APEX evaluation uses CFG 1.5 and 250 Euler steps.

Table 4: Training compute to match the published f8c4 SiT-XL gFID on ImageNet-512. Batch sizes are global. Training FLOPs use Equation 10.

Latent	Tokens	Steps	Batch	Training FLOPs	gFID
SD-VAE f8c4	1,024	3,000,000	256	1.21×10^{21}	2.62
APEX f64c8$-$32$-$128	64	1,000,000	2,048	1.78×10^{20}	2.62

C AUTOENCODER TRAINING

We train all autoencoders on ImageNet (Deng et al., 2009) with a global batch size of 64. We follow the DC-AE architecture, training recipe, and published implementation (Chen et al., 2025c), including its optimization settings and reconstruction, perceptual (Zhang et al., 2018), and adversar-

ial losses. APEX adds channel-prefix sampling and resolution-aware supervision; the encoder and decoder remain shared across all levels.

At each training step, we sample a cumulative level uniformly, retain its channels, and set all later channels to zero. The decoder produces a full-resolution image. We then resize both the reconstruction and the target to the selected level's resolution before computing the loss.

Channel levels and target resolutions. Channel counts are cumulative: prefixes of 8, 32, and 128 channels add 8, 24, and 96 channels on the same spatial grid. We use the same structure-to-detail heuristic as in Section 3.1: at lower widths, a roughly $4\times$ increase in cumulative channels corresponds to a $2\times$ increase in target side length. At larger widths, we introduce intermediate levels to avoid excessively large refinement groups and treat this proportionality as a guideline. Exact assignments are given in Table 5.

Table 5: APEX hierarchies, target side lengths, and main SiT-XL/1 clock settings for ImageNet-512. The gap and training allocation are defined in Section D.

Latent	Cumulative channels (group sizes)	Target side lengths	g	p_{L_0}
f64c128	8–32–128 (8, 24, 96)	128, 256, 512	0.25	0.9
f64c256	8–32–128–256 (8, 24, 96, 128)	128, 256, 512, 512	0.25	0.9
f64c512	8–32–128–256–512 (8, 24, 96, 128, 256)	128, 256, 512, 512, 512	0.25	0.9
f128c512	32–128–256–512 (32, 96, 128, 256)	128, 256, 512, 512	0.25	0.9
f128c1024	32–128–256–512–1024 (32, 96, 128, 256, 512)	128, 128, 256, 256, 512	0.25	0.9

Full-resolution control. The full-resolution control keeps the 8–32–128 channel hierarchy and prefix sampling, but supervises every prefix at 512×512 (row (e) of Table 2).

Algorithm 1 shows the training loop, with APEX changes in green. The base loss is the DC-AE objective for the current training stage.

Algorithm 1. Autoencoder training

Standard autoencoder

```
for images in data:
  z = encoder(images)
  pred = decoder(z)
  loss = base_loss(pred, images)
  update(loss)
```

APEX

```
for images in data:
  z = encoder(images)
  level = uniform_level()
  z = keep_prefix(z, level)
  pred = decoder(z)
  size = target_size[level]
  pred = resize(pred, size)
  target = resize(images, size)
  loss = base_loss(pred, target)
  update(loss)
```

`keep_prefix` zeros all channels after the sampled level, leaving the decoder input at full width. The loss is the image-space objective on `pred` and `target`; the zeroed channels are not supervised.

Reconstruction quality. We train the f64c128 and f128c512 autoencoders to match the rFID of the corresponding official DC-AE models.

D Staggered Training and Inference

D.1 The staggered clock

The global clock runs from $\tau = 0$ to $T = 1 + (L-1)g$, where L is the number of levels and g is the stagger gap. Each level has a local diffusion time

$$t_\ell(\tau) = \mathrm{clip}(1 + \ell \cdot g - \tau, 0, 1). \tag{11}$$

Local time decreases from 1 (noise) to 0 (clean). Level ℓ starts at $\tau = \ell \cdot g$ (a multiple of the gap) and finishes at $\tau = 1 + \ell \cdot g$. Earlier levels are therefore cleaner than the later levels they condition.

With $g = 0.25$, the three denoising windows are $(0, 1)$, $(0.25, 1.25)$, and $(0.5, 1.5)$ (Figure 11). Later levels begin before the earlier levels have finished.

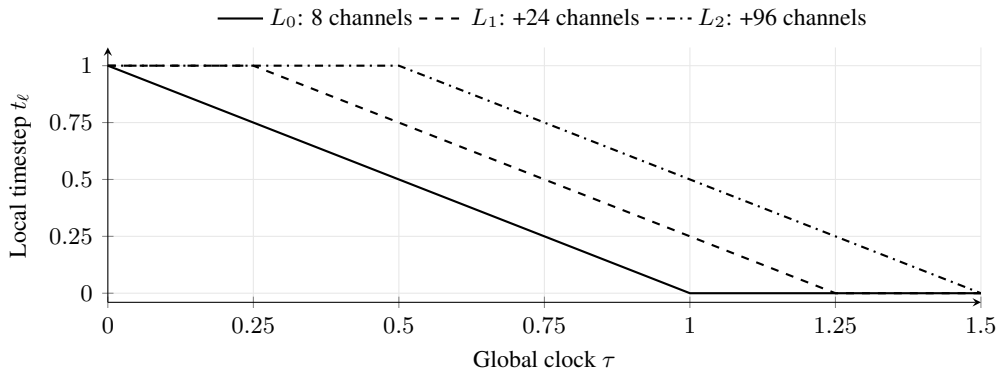


Figure 11: Three-level staggered clock with $g = 0.25$. Before a level starts, its input is zero. During the descending segment it is denoised. Once its local time reaches zero, it stays fixed as context for the remaining levels.

D.2 Training-clock sampling

We place probability p_{L_0} on the interval $(0, 1)$, where L_0 is still denoising, and the remaining probability on $(1, T)$. Sampling is uniform within each interval:

$$q(\tau) = \begin{cases} p_{L_0}, & 0 < \tau < 1, \\ \dfrac{1 - p_{L_0}}{T - 1}, & 1 < \tau < T. \end{cases} \tag{12}$$

Uniform sampling over the full clock corresponds to $p_{L_0} = 1/T$. When $g = 0$, the whole clock is $(0, 1)$ and all levels share the same timestep.

Most training examples are drawn while L_0 is active because it must learn structure generation. Refinement levels also receive training during the parts of their trajectories that overlap this interval. For example, with three levels, $g = 0.25$, and $p_{L_0} = 1$, L_1 is trained at local times $(0.25, 1)$ and L_2 at $(0.5, 1)$. Inference still traverses their complete trajectories.

Main-model clock settings are listed in Table 5.

D.3 Diffusion training

Unstarted groups are zero, active groups are mixed with Gaussian noise, and completed groups provide clean context. The target velocity is noise minus the clean latent. We average the squared error within each active group and then equally across active groups. Unstarted and completed groups do not contribute to the loss.

The input-projection weights for L_0 use the standard SiT initialization (Ma et al., 2024); those for later groups are initialized to zero. The Transformer is conditioned on the global clock.

Algorithm 2. Diffusion training

Standard diffusion

```
for images, labels in data:
  z = encode(images)
  noise = gaussian_like(z)
  time = sample_time()
  noisy = mix(z, noise, time)
  pred = model(noisy, time, labels)
  target = noise - z
  loss = mse(pred, target)
  update(loss)
```

APEX

```
for images, labels in data:
  z = encode(images)
  noise = gaussian_like(z)
  clock = sample_clock()
  times = local_times(clock)
  noisy = prepare_groups(
      z, noise, times)
  pred = model(noisy, clock, labels)
  target = noise - z
  loss = active_group_mse(
      pred, target, times)
  update(loss)
```

`sample_clock` draws $\tau \sim q$. `mix` forms $(1-t)z + t\epsilon$. `prepare_groups` sets the three group states described above; `active_group_mse` averages their active losses. Class dropout is unchanged.

D.4 Diffusion inference

Each group is initialized from Gaussian noise when its window opens. One shared prediction updates all groups using their local-time changes:

$$z_\ell \leftarrow z_\ell + \big[t_\ell(\tau_{k+1}) - t_\ell(\tau_k)\big] v_\ell. \tag{13}$$

Unstarted and completed groups have zero time change and remain fixed. The grid includes the activation points $\ell \cdot g$ (multiples of the gap), where new groups are initialized before the model evaluation.

Algorithm 3. Diffusion inference

Standard diffusion

```
z = gaussian_latent()
for time, next_time in time_grid:
  pred = predict(z, time, label)
  delta = next_time - time
  z = z + delta * pred
image = decode(z)
```

APEX

```
z = zeros_latent()
for clock, next_clock in grid:
  start_new_groups(z, clock)
  pred = predict(z, clock, label)
  delta = local_times(next_clock)
  delta = delta - local_times(clock)
  z = z + delta * pred
image = decode(z)
```

`predict` combines the conditional and null-class predictions as $v = v_{\text{null}} + s(v_{\text{class}} - v_{\text{null}})$ when guidance is used. Setting $s = 1$ gives unguided class-conditional sampling. `start_new_groups` initializes each group once. The completed latent is decoded by the shared autoencoder.

Prefix evaluation. For Figure 8(c), we redistribute 250 NFEs over $(0, 1)$, $(0, 1.25)$, or $(0, 1.5)$ and decode the completed prefix, zeroing later channels. These intervals complete one, two, or three levels at $g = 0.25$.

E Complete Numerical Results

The tables below give the values in the main figures and additional generation metrics. Each row contains one model or experimental setting. Unguided gFID is class-conditional sampling at CFG 1.0. It is reported in Tables 6 and 7 for every setting with a matched guided run, and for the SiT-B gap and clock-allocation ablations. A dash marks a setting whose guided run was not identified. sFID and IS in Tables 6 and 7 are computed on the same unguided samples. The remaining tables report guided gFID, and their sFID and IS use the guided samples.

E.1 GENERATION ACROSS CHANNEL COUNTS

Table 6: Figure 5(a): generation at f64. SiT-XL/1 models use 500K training steps and batch size 2,048. Unguided gFID, sFID, and IS use CFG 1.0. A dash denotes a metric that was not directly evaluated.

		gFID ↓		sFID ↓	IS ↑
Channels	Method	w/o CFG	CFG 1.5	w/o CFG	w/o CFG
128	DC-AE	13.45	3.22	6.40	106
128	DC-AE 1.5	13.13	3.15	6.3	109
128	FreqWarm	15.61	3.53	6.92	100
128	Spectrum Matching	13.96	3.41	6.04	99
128	**APEX**	**10.94**	**2.85**	**6.05**	**121**
256	DC-AE	30.55	9.19	9.33	58
256	DC-AE 1.5	22.77	7.11	7.5	69
256	FreqWarm	24.96	7.54	7.5	64
256	Spectrum Matching	29.35	8.28	7.44	57
256	**APEX**	**14.82**	**3.15**	**7.42**	**108**
512	DC-AE	54.25	26.10	13.31	31
512	DC-AE 1.5	37.85	14.76	10.8	60
512	FreqWarm	46.02	20.80	12.1	45
512	Spectrum Matching	54.11	26.4	10.92	70
512	**APEX**	**22.49**	**4.90**	**8.57**	**86**

Table 7: Figure 5(b): generation at f128. SiT-XL/1 models use 1M training steps and batch size 4,096. Unguided gFID, sFID, and IS use CFG 1.0.

		gFID ↓		sFID ↓	IS ↑
Channels	Method	w/o CFG	CFG 1.5	w/o CFG	w/o CFG
512	DC-AE	41.01	14.44	11.91	49
512	DC-AE 1.5	36.96	11.86	12.55	57
512	FreqWarm	39.54	13.57	11.75	50
512	Spectrum Matching	40.63	13.77	12.95	51
512	**APEX**	**31.01**	**8.98**	**14.49**	**65**
1024	DC-AE	115.74	89.47	32.97	13
1024	DC-AE 1.5	84.92	51.47	28.12	26
1024	FreqWarm	109.09	79.51	28.15	14
1024	Spectrum Matching	107.74	73.71	34.33	15
1024	**APEX**	**48.56**	**18.77**	**18.98**	**42**

E.2 SCHEDULE, COMPONENT, AND HIERARCHY ABLATIONS

The Figure 8 and component ablations use SiT-B/1 at f64c128. Training FLOPs are matched within each comparison. Guided results use CFG 1.5. Unguided gFID is reported for the gap and clock-allocation comparisons.

Table 8: Figure 8(a): stagger gap. † extends the main plot.

	gFID ↓		sFID↓	IS↑
Gap g	w/o CFG	CFG 1.5		
0.00	50.01	21.08	8.64	87
0.10	40.10	13.73	7.75	120
0.25	31.38	8.52	7.32	160
0.50	33.48	9.35	10.00	166
0.75†	45.40	16.66	14.34	130

Table 9: Figure 8(b): training allocation at $g = 0.25$.

	gFID ↓		sFID↓	IS↑
p_{L_0}	w/o CFG	CFG 1.5		
$2/3$	35.45	10.87	7.09	137
0.9	31.38	8.52	7.32	160
1	30.77	8.12	6.78	163

$p_{L_0} = 2/3$ is uniform clock sampling.

Table 10: Figure 8(c): completed prefixes of the $p_{L_0} = 1$ model.

Channels	gFID ↓ CFG 1.5	sFID↓	IS↑
8	19.83	19.76	88
32	9.04	8.89	150
128	8.12	6.78	163

The clock stops at 1, 1.25, and 1.5 for the first one, two, and three levels, respectively. Each row redistributes all 250 NFEs over that interval.

Table 11: Additional metrics for the component study in Table 2.

Configuration	gFID ↓ CFG 1.5	sFID↓	IS↑
(a) DC-AE / synchronous	13.27	7.42	117
(b) APEX AE / channel mean	20.37	7.69	86
(c) APEX AE / level mean	21.08	8.64	87
(d) APEX AE / prefix sampling	17.83	10.50	109
(e) Full-res. AE / stagger	8.77	6.87	161
(f) DC-AE 1.5 / stagger	52.89	11.48	31
(g) Full APEX	8.12	6.78	163

F Figure Reproduction

This section describes how we produce each figure. Training and evaluation follow Section B; channel hierarchies and the staggered process are given in Sections C and D.

F.1 Figure 1: overview

Purpose. Show how APEX organizes a wide latent, denoises its levels asynchronously, and changes the reconstruction–generation trade-off.

Procedure. Panel (a) is a schematic of an f128c512 latent: a 512^2 image is encoded onto a 4×4 grid of 16 tokens, and the 512 channels of each token are ordered from structure to detail. Panel (b) shows the corresponding staggered schedule on one global clock τ: each later level starts after a delay g, so it is denoised while earlier levels are already cleaner. Panel (c) is a schematic of the reconstruction–generation frontier relative to a flat latent; the points and curves are illustrative, not measured. The quantitative five-method comparison at f128c512 is Figure 5.

F.2 Figure 2: autoencoder, training, and inference

Purpose. Show how APEX changes autoencoder training, diffusion training, and sampling.

Procedure. The autoencoder panel encodes an image, samples a cumulative prefix, zeros the remaining channels, and decodes at full resolution. Reconstruction and target are then resized to the prefix's target resolution before the loss. The example retains the first two of four levels. The training panel samples a global clock from $q(\tau)$ and maps it to local times with Equation 11. With $g = 0.25$ and $\tau = 0.38$, the first two local times are 0.62 and 0.87; the third level has not started. Active groups are noised at these times, the unstarted group is zeroed, and one shared Transformer predicts their velocities. The inference panel shows the same three-level clock at the activation points: at $\tau = 0$ only L_0 is active; at $\tau = g$ and $\tau = 2g$, L_1 and L_2 are initialized from Gaussian noise. Generation continues until every group is clean.

F.3 Figure 3: staged generation

Purpose. Show the image after each level reaches $t = 0$.

Procedure. We run the three-level f64c8–32–128 model and decode the completed prefix when each successive level finishes. The displayed prefixes contain 8, 32, and 128 channels. Within each

example, all three images come from the same generation, and the inset shows the same crop. The difference visualizations in Section A.2 subtract successive decodes of these two examples.

F.4 Figure 4: training-clock density

Purpose. Show how training examples are allocated along the global clock.

Procedure. We use a three-level clock with $g = 0.25$, so its endpoint is $T = 1.5$. Uniform clock sampling has density $1/T = 2/3$ throughout the interval. For the structure-heavy example, we assign probability 0.9 to $(0, 1)$, where L_0 is active, and probability 0.1 to $(1, 1.5)$. Sampling uniformly within each interval gives density 0.9 in the first interval and 0.2 in the second. The figure plots these two densities; it does not require training another model.

F.5 Figure 5: generation across channel counts

Purpose. Compare generation as the latent becomes wider at fixed spatial compression and training compute.

Procedure. We compare DC-AE, DC-AE 1.5, FreqWarm, Spectrum Matching, and APEX at f64c128, f64c256, f64c512, f128c512, and f128c1024. Autoencoders are trained on ImageNet with the corresponding method's representation objective. For APEX, full-latent reconstruction quality is matched to the corresponding DC-AE model. We then train a SiT-XL/1 denoiser for each method and latent shape. The f64 runs use 500K steps at batch size 2,048, and the f128 runs use 1M steps at batch size 4,096, giving approximately the same cumulative training FLOPs.

We generate 50,000 class-conditional images per model with CFG 1.5 and compute gFID using the ADM protocol. Within each panel, the spatial grid stays fixed and only the channel count changes. The f64 panel uses 64 tokens and the f128 panel uses 16. Tables 6 and 7 give the numerical values, including unguided gFID at CFG 1.0 for every setting whose guided checkpoint was identified.

F.6 Figure 6: inference compute and guidance

Purpose. Test whether stronger spatial compression improves generation when both training and inference compute are limited.

Procedure. We train DC-AE f32c32 for 250K steps at batch size 1,024, and DC-AE f64c128 and APEX f64c8–32–128 for 500K steps at batch size 2,048. These schedules match denoiser-training compute. We then keep each trained model fixed and change its number of sampling steps. Every NFE and guidance setting is evaluated using 50,000 generated images.

For panel (a), we plot guided gFID against relative denoiser inference FLOPs. One f32 evaluation costs approximately four f64 evaluations, so the $1\times$ budget pairs 64 f32 NFEs with 250 f64 NFEs. The f32 curve uses 16, 32, 64, 128, and 250 NFEs. The f64 curves use 32, 64, 128, 250, 500, and 1,000 NFEs. At each inference budget, we sweep CFG in increments of 0.1 independently for each model, extending the sweep until gFID increases on both sides of the minimum. We report the best evaluated value.

For panel (b), we hold inference compute at $1\times$ and plot gFID against the guidance scale. At this budget, the resulting sweeps span 1.2–1.5 for f32 and 1.3–1.6 for the f64 models. APEX and f32 select 1.4, while DC-AE f64 selects 1.5.

F.7 Figure 7: prefix reconstructions

Purpose. Show the information retained by early prefixes.

Procedure. We encode the same image with DC-AE 1.5 and APEX. We retain a cumulative prefix, set all later channels to zero, and decode at full resolution. The DC-AE 1.5 examples use 16, 64, and 256 channels; APEX uses the resolution-aware hierarchy shown in the figure. We display the same image region at each level to compare the detail already present in the early prefixes.

F.8 FIGURE 8(A): STAGGER GAP

Purpose. Measure the trade-off between cleaner conditioning and overlap between levels' denoising windows.

Procedure. We keep the resolution-aware f64c8–32–128 autoencoder, SiT-B/1 backbone, training-clock allocation, and training FLOPs fixed, and train with $g \in \{0, 0.1, 0.25, 0.5\}$. At $g = 0$, every group has the same timestep, while the loss remains equally weighted across groups. For each gap, we sample over the complete global clock with the same NFE budget and CFG 1.5, then compute gFID from 50,000 images. Thus a larger gap changes the overlap and clock length without increasing the sampling budget. Table 8 also reports the $g = 0.75$ extension.

F.9 FIGURE 8(B): TRAINING ALLOCATION

Purpose. Measure how concentrating training on structure generation affects quality.

Procedure. We keep the same three-level autoencoder and fix $g = 0.25$. We train three SiT-B/1 models at the same compute budget, changing only the clock-sampling distribution: uniform over the full clock, $p_{L_0} = 0.9$, and $p_{L_0} = 1$. For uniform sampling, $p_{L_0} = 2/3$. The other two settings sample within the intervals using Equation 12. All three models are evaluated with the same sampling budget, CFG 1.5, and 50,000 generated images.

F.10 FIGURE 8(C): CONTRIBUTION OF SUCCESSIVE LEVELS

Purpose. Test whether later levels improve generation beyond the structural prefix.

Procedure. We use the same $p_{L_0} = 1$ model for all three conditions. We stop the global clock at 1, 1.25, or 1.5, when the first one, two, or three levels have finished. In every case, we redistribute all 250 NFEs over the retained interval. We then zero the channels after the completed prefix and decode its 8, 32, or 128 channels with the same autoencoder. Each condition produces 50,000 images with CFG 1.5. The resulting gFIDs are 19.83, 9.04, and 8.12. No additional diffusion models are trained for this comparison.

F.11 TABLE 2: COMPONENT ANALYSIS

Purpose. Separate the effects of the representation, loss, and asynchronous training.

Procedure. We train all seven variants with SiT-B/1 at f64c128 and the same cumulative training FLOPs. Row (a) uses the flat DC-AE latent, one shared timestep, and a loss averaged over all channels. Row (b) applies the same training procedure to the APEX autoencoder. Row (c) keeps the full latent and shared timestep, but averages the loss within each channel group and then equally across groups. Row (d) instead samples a cumulative prefix uniformly, zeros the suffix, and noises the retained groups at one shared timestep; its loss is averaged equally over those groups.

Row (e) keeps the 8–32–128 channel partition and APEX diffusion, but trains every autoencoder prefix against the full-resolution target. Row (f) uses the DC-AE 1.5 hierarchy with 16, 20, ..., 128 channels and uniform cumulative-prefix sampling, with staggered local times on the retained groups. Row (g) uses the complete resolution-aware APEX hierarchy and diffusion procedure. We evaluate each model using the common generation protocol.

F.12 ADDITIONAL VISUALIZATIONS AND COMPARISONS

Purpose. Show progressive generation and refinement differences.

Procedure. For Figure 9, we sample from the staggered generator: eight examples with f64c8–32–128, four with f64c8–32–128–256, and four with f64c8–32–128–256–512. When each successive level reaches $t = 0$, we decode the completed prefix with the shared autoencoder. These are generations, not reconstructions of real images.

For Figure 10, we subtract consecutive full-resolution prefix decodes before resizing or adding the inset crops. Both examples and both differences use the same display scale.

Figure 11 evaluates Equation 11 for three levels and $g = 0.25$.