PLAIN FLOW MAPS WITH ANALYTIC HEADS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow maps learn direct jumps between arbitrary points along diffusion ODE trajectories via differential objectives. This requires differentiating the entire model with respect to time, incurring severe overheads. We show that this is unnecessary. We introduce plain flow maps, which retain a standard diffusion backbone and expose the differentiated time only to a lightweight analytic head. We show that this head can be as simple as a single layer featuring time-adaptive spatial and channel mixing, yet remains highly effective. We introduce two parameterizations, secant and velocity, allowing terminal-time differentiation to be resolved solely within the head via closed-form derivatives and integrals, respectively. This design completely bypasses explicit differentiation tools and their associated drawbacks. Extensive experiments demonstrate remarkable performance on from-scratch ImageNet and text-to-image training.

1 INTRODUCTION

Diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020) have emerged as the state-of-the-art paradigm for generative modeling, achieving unprecedented success in generating high-fidelity images (Dhariwal & Nichol, 2021; Rombach et al., 2022), videos (Blattmann et al., 2023b; Ho et al., 2022; Blattmann et al., 2023a), and audio (Kong et al., 2020; Huang et al., 2023; Liu et al., 2023). However, their reliance on iterative sampling over numerous steps fundamentally bottlenecks their inference speed.

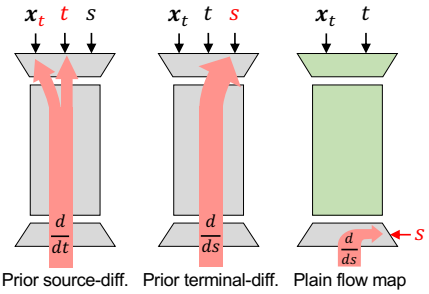


Figure 1: **Comparison of differentiation scopes across different flow-map formulations.** Unlike prior source- and terminal-differentiation baselines that force explicit differentiation through the entire deep backbone, our plain flow maps strictly confine all explicit calculus to a lightweight analytic head. This isolates temporal dynamics from heavy spatial representation learning, leaving the massive backbone completely differentiation-free.

Flow maps (Boffi et al., 2024) address this limitation by learning to predict direct jumps from a source time to a desired terminal time along the diffusion Ordinary Differential Equation (ODE) trajectory. They have gained popularity due to their elegant principles and deterministic noise-data pairing that protects generative diversity. Despite the theoretical appeal, optimizing flow maps relies heavily on differential objectives, *i.e.*, source (Song et al., 2023; Kim et al., 2023; Geng et al., 2025) or terminal (Gu et al., 2023; Zhou et al., 2025b) differentiation. These formulations requires calculating the derivative of the model output with respect to a certain time variable. In practice, this time derivative is computed by exact Jacobian-Vector Products (JVP) using differentiation tools (e.g., `torch.jvp`) or approximated via finite differences. However, they introduce either inherent instability and low computational efficiency demanding extra engineering optimization, or inevitable discretization errors and numerical instability.

Recent evidence, however, suggests a potential way out. Lee et al. (2026) find that terminal-time conditioning can be delayed to shallow late-stage blocks with little loss in performance, although their reliance on source differentiation means the heavy full-backbone differentiation remains inevitable. Meanwhile, recent large-scale few-step distillation methods recover future trajectory information through lightweight output-side mechanisms (Chen et al., 2026; Yang et al., 2026; Shaul et al., 2026), implying that much of the future dynamics may already be encoded in the source

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



Figure 2: Selected samples of one-step class-conditional generation on ImageNet 256×256 (top two rows) and few-step text-to-image generation (bottom two rows).

representation. These observations motivate a radical leap. We therefore ask, *with a terminal-differentiated objective, can we completely strip the terminal time from the backbone?* In doing so, we could compute the required calculus lightweightly or even analytically within the head, bypassing the burdensome differentiation nature. We show that the answer is yes.

We begin by investigating the internal mechanics of terminal differentiation through two exploratory studies. We observe that even when the terminal time is injected from the very beginning following standard convention, its derivative contribution remains heavily concentrated in the later network stages. Furthermore, training the model from scratch with altered injection positions yields no significant difference in FID performance. Motivated by these insights, we introduce *plain flow maps*, a simplified architecture that elegantly circumvents the explicit differentiation bottleneck of flow-map training. Specifically, we retain a standard diffusion backbone, which extracts rich representations, and expose the differentiated terminal time exclusively to a lightweight analytic head. Under the terminal-differentiation paradigm, this framework naturally supports two fundamental trajectory parameterizations: the *secant* (the trajectory jump) and the *velocity*. While the secant formulation requires only taking the derivative of the head, allowing the use of any arbitrary nonlinear architecture, the velocity formulation necessitates computing a closed-form integral over time, which strictly limits the allowable function class. To seamlessly unify both parameterizations under a controlled architecture while maintaining sufficient representational capacity, we carefully design a mathematically tractable bilinear head. Specifically, we construct the head using a harmonic time space coupled with purely linear spatial and channel mixing structures. Unlike conventional designs that rely on log-spaced time and softmax attention, they avoid intractable time-dependent nonlinearities, guaranteeing exact analytic integrability and differentiability. Empirically, this analytic head proves remarkably effective despite its minimal parameter count.

Extensive experiments validate the effectiveness and scalability of plain flow maps. By completely bypassing the bottlenecks of explicit full-model differentiation, our approach dramatically simplifies the training process. On ImageNet-256×256, we achieve a 1-step FID of 2.89 and a 2-step FID of 2.40 with a standard DiT-XL/2 backbone, while significantly improving training efficiency over

other approaches. Scaling further to from-scratch text-to-image generation, our framework yields GenEval scores of 0.80 and 0.83 at 2 and 4 steps, respectively. Ultimately, plain flow maps provide a mathematically transparent, highly efficient, and easily scalable solution for generative modeling.

2 PRELIMINARY

We adopt the flow-matching (Liu et al., 2022; Lipman et al., 2022; Albergo et al., 2023) formulation with linear interpolants. Let $X_0 \sim P_0$ the data distribution and $X_1 \sim P_1 = \mathcal{N}(0, I)$ denote the base noise. The interpolant is

$$X_t = (1-t)X_0 + tX_1, \qquad t \in [0,1], \tag{1}$$

whose probability flow ODE is governed by the optimal velocity field

$$\frac{dX_t}{dt} = v^*(X_t, t), \qquad v^*(x,t) = \mathbb{E}[X_1 - X_0 \mid X_t = x]. \tag{2}$$

Rather than numerically integrating the instantaneous velocity field, a flow map (Boffi et al., 2024) directly models the transition between arbitrary times. We denote the flow map from source time t to terminal time s by $F_{t,s}(x)$, such that along the probability-flow trajectory,

$$F_{t,s}(X_t) = X_s, \qquad F_{t,t}(x) = x. \tag{3}$$

Flow maps satisfy the composition property

$$F_{s,r}\left(F_{t,s}(x)\right) = F_{t,r}(x), \tag{4}$$

allowing the same model to generate with arbitrary numbers of finite-time jumps.

The flow map admits two complementary differential characterizations (Boffi et al., 2024). The Eulerian form differentiates the source time while keeping the input location fixed:

$$\partial_t F_{t,s}(x) + \nabla_x F_{t,s}(x)\, v^*(x,t) = 0. \tag{5}$$

This gives the source-differentiation objective (Song et al., 2023; Kim et al., 2023; Geng et al., 2025)

$$\mathcal{L}_{\text{source}}(\theta) = \mathbb{E}\left[\left\|\partial_t F_\theta(X_t,t,s) + \nabla_x F_\theta(X_t,t,s)\, v^*(X_t,t)\right\|_2^2\right]. \tag{6}$$

In contrast, Lagrangian form differentiates the terminal time:

$$\partial_s F_{t,s}(x) = v^*\left(F_{t,s}(x), s\right), \tag{7}$$

leading to the terminal-differentiation objective (Gu et al., 2023; Zhou et al., 2025b)

$$\mathcal{L}_{\text{terminal}}(\theta) = \mathbb{E}\left[\left\|\partial_s F_\theta(X_t,t,s) - v^*\left(F_\theta(X_t,t,s), s\right)\right\|_2^2\right]. \tag{8}$$

3 PLAIN FLOW MAPS

In the source-differentiation objective (Eq. 6), the derivative depends on the spatial input X_t. Consequently, altering the position of time injection does not reduce the depth of the required differentiation graph. In contrast, the terminal-differentiation objective (Eq. 8) computes the gradient exclusively with respect to the terminal time. This strictly confines the explicit differentiation process to the network modules executed after the injection point.

Motivated by this structural advantage, we first conduct two exploratory studies demonstrating that terminal-time injection can be significantly delayed without performance degradation. Building on this insight, we propose *plain flow maps*. We present their general formulation and introduce the secant and velocity parameterizations. Next, we detail the design of the analytic head and the overall training objective. Finally, we discuss the relationship between our approach and closely related works.

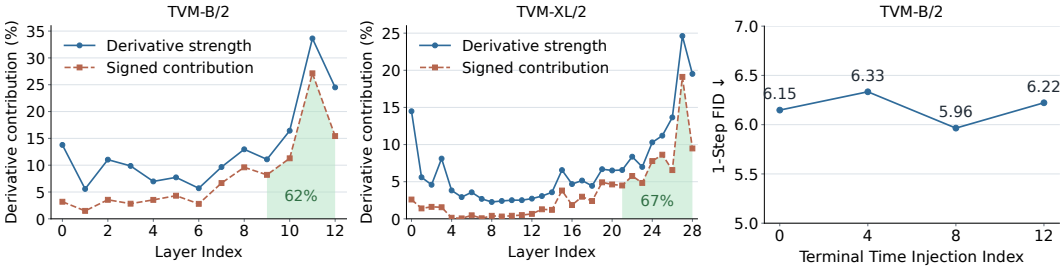


Figure 3: Layer-wise terminal derivative contributions for pretrained TVM-B/2 (left) and TVM-XL/2 (right). The effective terminal derivative is concentrated within the deeper layers.

Figure 4: The terminal time injection point has little effect on one-step FID performance.

3.1 Motivating Study on Terminal-Time Conditioning

We initially evaluate how terminal derivative contributions are distributed across a pretrained TVM (Zhou et al., 2025b) model. We then systematically delay the terminal time injection, retrain the models, and observe the resulting impact on generation quality. In the studies, we use $i \in \{0, 1, \ldots, L-1\}$ to index the L Transformer blocks of the TVM, and use $i = L$ to index the final linear layer. Other details are provided in Appendix A.

Terminal derivative contributions across blocks. We compute the exact contribution of each layer to the output terminal derivative through a chain-rule decomposition. We report both the derivative strength of each layer and its signed contribution to the final derivative in Figure 3. On both TVM-B/2 and TVM-XL/2, the effective terminal derivative is strongly concentrated in the later layers. Specifically, the final quarter of Transformer blocks together with the final linear layer account for 62% and 67% of the signed terminal contribution in TVM-B/2 and TVM-XL/2, respectively. Interestingly, several shallow blocks exhibit non-negligible derivative strength but a substantially smaller signed contribution. This suggests that much of their terminal-dependent response is subsequently canceled or redirected by downstream computation. Conversely, the strength and signed contribution become increasingly aligned in deeper layers, indicating that late-stage terminal computation contributes more directly to the final output derivative. This observation confirms that while terminal conditioning is present throughout the backbone, its functional effect is dominated by the later stages of the network.

Impact of delaying terminal time injection. We explore whether this terminal-time conditioning can be delayed without compromising generative performance. We alter the injection index i (meaning that the terminal time is injected before the i-th layer) and retrain the TVM-B/2 model, and present the results in Figure 4. Interestingly, we observe that the choice of injection index i has a negligible impact on one-step FID performance. Notably, the optimal generative quality is not even achieved at $i = 0$, and delaying the injection to $i = 12$ (where the terminal time bypasses all Transformer blocks and exclusively enters the head) results in negligible degradation, confirming that early terminal-conditioned computation is not necessary for strong generation performance.

Together, these results reveal a consistent picture: terminal-time dependence is functionally concentrated in the later stages of the network, and the remaining terminal-dependent computation can be shifted entirely toward the output without sacrificing generative quality. This strongly motivates us to remove terminal time from the Transformer backbone altogether and confine it to a lightweight head, thereby enabling fully analytic terminal differentiation while preserving the expressive capacity of the underlying diffusion backbone.

3.2 Plain Backbone with an Analytic Head

Let $z_t = g_\phi(x_t, t)$ denote the representation produced by a standard source-time diffusion backbone. The terminal time s is never exposed to the backbone and is introduced only through a lightweight head. We define

$$f_\theta(x_t, t, s) = h_\psi(z_t, s), \qquad z_t = g_\phi(x_t, t), \tag{9}$$

where $\theta = (\phi, \psi)$ collectively denotes the model parameters.

The corresponding flow map is

$$F_\theta(x_t, t, s) = x_t + \mathcal{T}_{t,s}\left[f_\theta(x_t, t, \cdot)\right], \tag{10}$$

where $\mathcal{T}_{t,s}$ converts the predicted trajectory quantity f_θ into the finite-time displacement from t to s.

Since the terminal time appears only in the head, $\partial_s g_\phi(x_t, t) = 0$, and all terminal-time differentiation is confined to h_ψ. We consider two choices of $\mathcal{T}_{t,s}$, corresponding to secant and velocity parameterizations.

Plain-Secant parameterizes $f_\theta(x_t, t, s)$ as the average velocity along the trajectory between the source time t and terminal time s:

$$\frac{x_s - x_t}{s - t} = \frac{1}{s - t}\int_t^s v^*(x_r, r)\, dr. \tag{11}$$

The secant parametrization continuously recovers the velocity at the diagonal boundary:

$$\lim_{s \to t} \frac{x_s - x_t}{s - t} = v^*(x_t, t). \tag{12}$$

The transition operator becomes

$$\mathcal{T}^{\text{sec}}_{t,s}\left[f_\theta(x_t, t, \cdot)\right] = (s - t) f_\theta(x_t, t, s), \tag{13}$$

yielding the flow map

$$F^{\text{sec}}_\theta(x_t, t, s) = x_t + (s - t) f_\theta(x_t, t, s). \tag{14}$$

Differentiating with respect to the terminal time gives

$$\partial_s F^{\text{sec}}_\theta(x_t, t, s) = f_\theta(x_t, t, s) + (s - t)\partial_s f_\theta(x_t, t, s). \tag{15}$$

Since the terminal-time dependence of f_θ is confined to h_ψ, the terminal derivative only differentiates the analytic head.

Plain-Velocity parameterizes $f_\theta(x_t, t, s)$ as the instantaneous velocity along the trajectory at the terminal time s:

$$v^*(x_s, s). \tag{16}$$

At $s = t$, it directly recovers the instantaneous source velocity $v^*(x_t, t)$.

The corresponding transition operator is

$$\mathcal{T}^{\text{vel}}_{t,s}\left[f_\theta(x_t, t, \cdot)\right] = \int_t^s f_\theta(x_t, t, r)\, dr, \tag{17}$$

yielding the flow map

$$F^{\text{vel}}_\theta(x_t, t, s) = x_t + \int_t^s f_\theta(x_t, t, r)\, dr. \tag{18}$$

By the fundamental theorem of calculus,

$$\partial_s F^{\text{vel}}_\theta(x_t, t, s) = f_\theta(x_t, t, s). \tag{19}$$

By construction, the source representation $z_t = g_\phi(x_t, t)$ is independent of the integration variable r. The source state x_t determines the entire predicted future velocity curve through z_t, while r only queries the velocity at different future times. Consequently, the integration from t to s acts only on the analytic head and does not require re-encoding intermediate states along the trajectory.

3.3 Analytic Terminal Head

To preserve the property of analytically differentiable and integrable with respect to the terminal time while retaining sufficient modeling capacity, we use a harmonic time embedding, AdaLN-style adaptive channel mixing, and time-adaptive linear attention for spatial mixing. Figure 5 illustrates the overall head structure, and the complete analytic derivations are detailed in Appendix B.

Harmonic time representation. Unlike the standard diffusion timestep embedding, which uses sinusoidal features with log-spaced frequencies (Ho et al., 2020; Peebles & Xie, 2023), we use a

uniform harmonic frequency grid for analytic closure. Let $\delta = t - s$ denote the relative terminal time, K the number of harmonic frequencies, and ω_0 the fundamental frequency. We encode δ as

$$e(\delta) = [\cos(\omega_k \delta), \sin(\omega_k \delta)]_{k=1}^{K} \in \mathbb{R}^{2K}, \qquad \omega_k = k\omega_0. \tag{20}$$

This representation admits closed-form differentiation and integration. More importantly, the integer harmonic grid is closed under multiplication: products of harmonics remain finite sums of harmonics at integer sum-and-difference frequencies. Consequently, the time-adaptive operations introduced below can be composed while preserving exact temporal differentiation and integration.

Channel mixing. We follow the standard AdaLN (Perez et al., 2018) form to produce and apply channel-wise scale and shift. Since the scale and shift are linear functions of the harmonic basis, the resulting head remains analytically differentiable and integrable with respect to the terminal time. We refer to the variant restricted solely to channel mixing as the **linear head**.

Spatial mixing. Channel mixing acts independently on individual tokens. To additionally model terminal-time-dependent interactions across tokens, we introduce a lightweight linear-attention mechanism. Queries are computed solely from the source representation, while keys and values are modulated by channel-wise coefficients generated from the harmonic time embedding. We omit softmax and other time-dependent nonlinear normalization, so the attention output remains an integrable finite product of harmonic functions. Correspondingly, the variant incorporating this spatial attention mechanism is denoted as the **bilinear head**.

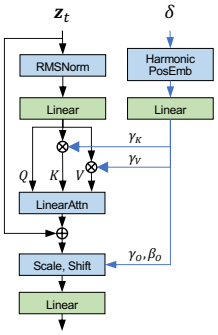


Figure 5: Overview of our blinear head architecture.

3.4 Training Objective

We combine the CFG-aware (Tang et al., 2025; Chen et al., 2025a) flow-matching loss with the terminal objective Eq. 8, similar to TVM (Zhou et al., 2025b). Let c denote the condition and w the CFG scale. Since both variants parameterize $f_\theta(x_t, t, t)$ as the velocity, we define

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}\left[\| f_\theta(X_t, t, t, c, w) - [w(X_1 - X_0) + (1-w) f_{\bar{\theta}}(X_t, t, t, \varnothing, 1)] \|_2^2\right] \tag{21}$$

and the complete from-scratch flow map matching objective

$$\mathcal{L}_{\mathrm{FMM}}(\theta) = \mathcal{L}_{\mathrm{FM}}(\theta) + \mathbb{E}\left[\| \partial_s F_\theta(X_t, t, s, c, w) - f_{\bar{\theta}}\left(\mathrm{sg}\left[F_\theta(X_t, t, s, c, w)\right], s, s, c, w\right) \|_2^2\right], \tag{22}$$

where sg denotes stop-gradient and $\bar{\theta}$ is the EMA model. Our parameterization further allows the local prediction $F_\theta(X_t, t, t, c, w)$ and the jump prediction $F_\theta(X_t, t, s, c, w)$ to share the same source representation $z_t = g_\phi(X_t, t, c, w)$. They differ only in the terminal-time query to the lightweight head. Consequently, both objectives require only a single backbone evaluation, eliminating the extra full-backbone pass demanded by TVM.

3.5 Relation to Prior Methods

Plain-Velocity is closely related to several recent trajectory-distillation methods that similarly encode the source state once and use lightweight mechanisms to predict future velocities. ArcFlow (Yang et al., 2026) uses a specialized mixture-of-momenta parameterization to construct a continuous integrable velocity curve, while PDD (Shaul et al., 2026) employs a collection of linear projectors to produce piecewise-constant mean velocities. π-Flow (Chen et al., 2026) follows a similar principle through a learned rollout policy, but additionally requires the evolving rollout state for each velocity query. Our plain flow maps provide a unifying framework where ArcFlow and PDD can be regarded as specific instances of Plain-Velocity, while π-Flow corresponds to a state-dependent variant. In contrast, Plain-Secant introduces a distinct secant-based parameterization that has no counterpart in these prior methods. Furthermore, unlike prior approaches, our implementation relies entirely on standard, non-specialized neural modules.

4 Experiments

To evaluate the effectiveness, scalability, and computational efficiency of plain flow maps, we conduct extensive experiments across diverse generative tasks. We first perform comprehensive abla-

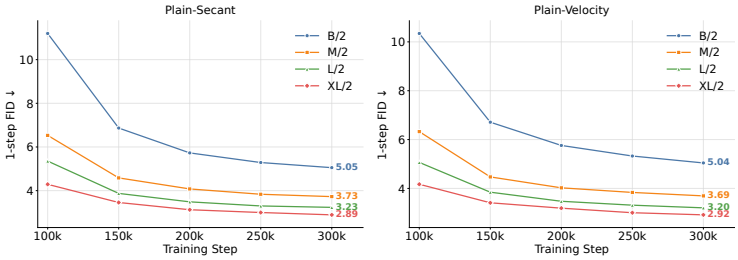


Figure 6: Scalability of Plain-Secant and Plain-Velocity.

Table 1: Effect of head design on one-step FID for the secant and velocity parameterizations.

Head	Secant	Velocity
DiT	5.39	N/A
linear	5.97	6.49
blinear	5.05	5.04

Table 2: Few-step diffusion models trained from scratch on class-conditional ImageNet-256 $\times$ 256.

Method	Params	NFE	FID ↓
iCT-XL/2 (Song & Dhariwal, 2023)	675M	1 / 2	34.24 / 20.30
Shortcut-XL/2 (Frans et al., 2024)	675M	1 / 4	10.60 / 7.80
IMM-XL/2 (Zhou et al., 2025a)	675M	1×2 / 2×2	7.77 / 3.99
MeanFlow-XL/2 (Geng et al., 2025)	676M	1 / 2	3.43 / 2.93
SoFlow-XL/2 (Luo et al., 2026)	676M	1 / 2	2.96 / 2.66
TVM-XL/2 (Zhou et al., 2025b)	678M	1 / 2	3.29 / 2.80
Plain-Secant-XL/2 (Ours)	679M	1 / 2	**2.89** / **2.40**
Plain-Velocity-XL/2 (Ours)	679M	1 / 2	2.92 / 2.43

Table 3: Comparison on training efficiency. † With TVM kernel; otherwise out of memory.

Method	Batchsize	Time (s)	Mem (G)
DiT-XL/2 (Peebles & Xie, 2023)	256	0.17	26.49
MeanFlow-XL/2† (Geng et al., 2025)	256	0.81	46.73
SoFlow-XL/2 (Luo et al., 2026)	256	0.26	27.04
TVM-B/2 (Zhou et al., 2025b)	256	0.31	19.37
TVM-XL/2 (Zhou et al., 2025b)	256	0.95	71.44
Plain-Secant-XL/2 (Ours)	256	0.24	27.60
Plain-Secant-XL/2 (Ours)	1024	0.72	68.45

tions and scaling studies on ImageNet-256×256 generation from scratch. To demonstrate broad applicability, we then extend our framework to text-to-image generation from scratch.

4.1 IMAGENET GENERATION FROM SCRATCH

Settings. To highlight the "plain" nature of our framework, we adopt the standard DiT (Peebles & Xie, 2023) as our backbone. We largely follow the training configurations of TVM (Zhou et al., 2025b), which serves as our primary terminal-differentiation baseline. Recognizing that the optimal classifier-free guidance (CFG) scale varies across different head architectures, we condition the model on the CFG scale during training, enabling an efficient search for the optimal scale during inference. Our training objective (Eq. 22) allows us to share gradient-tracking evaluations, necessitating computing the flow-matching loss at source time t instead of terminal time s. Since TVM's original sampling skews t towards 1, we adjust the gap parameter μ_g from -0.8 to -1.6 to center the time distribution, harmonizing both training objectives. For evaluation, we compute the Fréchet Inception Distance (FID) (Heusel et al., 2017) on 50K class-balanced samples. Further details are provided in the Appendix C. In the following sections, we first ablate our analytic head design, and subsequently investigate the scaling properties of both the secant and velocity variants. Finally, we discuss the training efficiency and stability.

Ablation on head architectures. We first ablate the impact of head design under both trajectory parameterizations. For the Plain-Secant formulation, the terminal derivative does not enforce strict integrability. Thus, we compare three variants: the standard DiT head (featuring non-linear AdaLN and SiLU), the linear head and the bilinear head. Conversely, for the Plain-Velocity formulation, the standard DiT head is strictly excluded due to its mathematical intractability for closed-form integration. Results in Table 1 indicate that the addition of spatial mixing significantly improves performance across both the secant and velocity parameterizations. Specifically, under the secant formulation, while the loss of nonlinearity causes the linear head to lag behind the original DiT head, the bilinear head closes this gap and even surpasses its highly non-linear counterpart. This proves that carefully designed bilinear spatial and channel mixing provides ample representational capacity while satisfying strict integrability constraints.

Generative performance and scaling property. As shown in Figure 6, both Plain-Secant and Plain-Velocity exhibit predictable scaling across standard DiT configurations from B/2 to XL/2, with FID scores monotonically decreasing as capacity expands. This confirms that uncoupling the terminal time from the backbone introduces no representational bottlenecks. At the XL/2 scale (Table 2), our method demonstrates superior generative quality. Plain-Secant-XL/2 achieves an

Table 4: Text-to-image generation results on GenEval (Ghosh et al., 2023) and DPG-Bench (Hu et al., 2024). See Appendix D for full metrics.

Method	Params	NFE	GenEval ↑	DPG-Bench ↑
Diffusion models				
PixArt-α (Chen et al., 2024)	0.6B	20×2	0.48	71.11
SD3-8B (Esser et al., 2024)	8B	50×2	0.68	-
FLUX.1-dev (Labs, 2024)	12B	50	0.67	83.32
DALL-E 3 (Betker et al., 2023)	-	-	0.67	83.50
OmniGen2 (Wu et al., 2025)	4B	50×2	0.80	83.57
PixelGen (Ma et al., 2026)	1.1B	25×2	0.79	80.01
Distilled few-step diffusion models				
SDXL-Turbo (Sauer et al., 2024b)	2.6B	8	0.50	-
SANA-Sprint (Chen et al., 2025c)	1.6B	8	0.72	81.10
SD3.5-Turbo (Sauer et al., 2024a)	8B	8	0.66	79.03
FLUX.1-Schnell (Labs, 2024)	12B	1	0.68	84.94
From-scratch few-step diffusion models				
TiM (Wang et al., 2026b)	0.9B	1	0.67	74.93
Plain-Secant (Ours)	1.1B	1	0.78	76.08
		2	0.80	77.65
		4	0.83	78.85
		8	0.84	80.11

FID of 2.89 / 2.40 at 1 / 2 steps, and Plain-Velocity-XL/2 yields a closely comparable 2.92 / 2.43. Selected qualitative samples from Plain-Secant-XL/2 are illustrated in Figure 2, with an extensive collection of uncurated results provided in Appendix E. We provide extended results in Appendix D. This demonstrates that confining the differentiation process to a lightweight head can achieve even better generative performance than prior full-backbone methods.

Training efficiency and stability. Table 3 illustrates the computational advantage of plain flow maps. Under the same terminal-differentiation objective, our XL/2 model is remarkably faster than the much smaller TVM-B/2 (0.24s vs. 0.31s). While SoFlow achieves comparable speed, it relies on finite differences that introduce inherent discretization errors; our formulation remains theoretically error-free through exact analytic computation. Furthermore, this efficiency translates into massive memory savings. Under a standard 80GB VRAM constraint, TVM-XL/2 maxes out at a batch size of 256; our framework effortlessly accommodates a batch size of 1024. Beyond computational savings, this design fundamentally alleviates the training instabilities associated with differentiating through deep architectures. Under the same objective, TVM suffers from unstable activations, forcing the introduction of intrusive architectural and weight-initialization modifications for semi-Lipschitz control. By confining exact analytic derivatives strictly to the head, our framework naturally stabilizes the optimization, allowing the plain DiT backbone to train robustly. Furthermore, this plain nature guarantees seamless compatibility with standard acceleration tools. While TVM requires custom-built Flash Attention kernels specifically engineered for JVPs, our method effortlessly leverages the standard Flash Attention used in conventional diffusion models.

4.2 Text-to-Image Generation from Scratch

Settings. To validate our framework beyond class-conditional generation, we train plain flow maps for text-to-image synthesis from scratch. We follow the efficient PixelGen (Ma et al., 2026) training recipe, pre-training on 36M images before fine-tuning on the BLIP3o-60K dataset (Chen et al., 2025b). We adopt the 1.1B-parameter PixelGen-XXL (Ma et al., 2026) alongside the FLUX.2 VAE, modify the input and output projections to match the VAE dimensions and remove the bottleneck deigned for pixel-space. We replace its output module with our analytic bilinear head and employ the secant parameterization, which shows better performance in our ImageNet experiments. The training objective and time sampling strategy remain identical to our ImageNet configuration. The CFG scale is conditioned on during training, and is set to 4 during inference. Further details are provided in the Appendix C.

Text-to-image generation performance. Table 4 evaluates our text-to-image generation capabilities on the GenEval (Ghosh et al., 2023) and DPG-Bench (Hu et al., 2024) benchmarks, demonstrating that our plain architecture achieves strong semantic alignment in the extreme low-step regime. At a single network evaluation, our Plain-Secant model achieves a GenEval score of 0.78. This outperforms existing from-scratch few-step approaches like TiM (Wang et al., 2026b) (0.67) and surpasses heavily parameterized distilled models, such as the 12B-parameter FLUX.1-Schnell (0.68), despite utilizing only 1.1B parameters. Furthermore, our framework also exhibits performance scalability with increased sampling steps. On the DPG-Bench, scaling the inference to 8 steps elevates the overall performance to 80.11, a score comparable to multi-step diffusion models. We present qualitative samples in Figure 2, with additional generated results provided in Appendix Figure 13. Ultimately, these results confirm that strictly confining explicit calculus to a lightweight head provides a highly efficient, from-scratch alternative to costly distillation pipelines without compromising complex text-image alignment.

5 Related Work

Diffusion and Flow Matching. Diffusion models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song & Ermon, 2019) generate samples by progressively reversing a noising process, while flow matching (Liu et al., 2022; Lipman et al., 2022; Albergo et al., 2023) learn continuous velocity fields that transport a simple prior to the data distribution. Sampling from these models requires numerical integration of the learned dynamics, typically involving many sequential network evaluations. This has motivated methods that directly learn finite-time transport along the underlying flow.

Few-Step ODE-Based Generative Models. Consistency Models (Song et al., 2023; Song & Dhariwal, 2023; Lu & Song, 2024) learn mappings from intermediate noisy states to trajectory endpoints. Consistency Trajectory Models (Kim et al., 2023) extend this concept to transitions between arbitrary timesteps, whereas MeanFlow (Geng et al., 2025) adopts an interval-averaged velocity parameterization of finite-time transport. Flow Map Matching (Boffi et al., 2024) and its self-distillation variants (Boffi et al., 2026) provide general frameworks for learning finite-time flow maps. Additional approaches construct flow maps via recursive bootstrapping (Frans et al., 2024) or differentiation-free Monte Carlo integration (Liu & Yue, 2025). Furthermore, Terminal Velocity Matching (TVM) (Zhou et al., 2025b) optimizes the flow map through a terminal-differentiated training objective. While these works primarily develop training objectives, their underlying time parameterizations typically cause explicit differentiation to unavoidably propagate through the entire backbone. Our work instead focuses on the flow-map parameterization itself. By fundamentally separating source encoding from terminal-time dependence, we ensure that terminal differentiation is strictly confined to a lightweight head.

6 Limitations and Future Work

Although our plain flow maps show strong effectiveness, generative quality at extremely low NFEs does not yet surpass the upper bound of full-step diffusion models. Furthermore, our current empirical validation focuses on from-scratch training. Future work will therefore explore integrating our architecture into trajectory-distillation pipelines to leverage pre-trained priors. Additionally, we plan to investigate alternative analytic head designs to further optimize learning dynamics.

7 Conclusion

We present plain flow maps, an efficient and elegant framework for few-step generative modeling. Fundamentally, under the terminal-differentiation objective, it decouples the generation process by conditioning the backbone solely on the current state and source time, while strictly confining the terminal time to the head. By confining all calculus to the head in closed-form, we leave the deep generative backbone completely free of explicit differentiation. This plain design circumvents the training instabilities and severe memory bottlenecks of full-backbone differentiation, and ensures seamless compatibility with standard off-the-shelf tools. Empirically, our method achieves strong generation quality in extreme low-step regimes on both ImageNet and text-to-image benchmarks.

AI USE STATEMENT

In this work, we used generative AI tools (such as large language models) for polishing the academic writing and assisting in drafting code scripts for experiments. We have not used generative AI tools for generating core research ideas, conceptualizing the proposed architecture, or deriving the fundamental mathematical proofs. All AI-assisted work was rigorously reviewed and validated. Specifically, all LLM-generated code was thoroughly verified, debugged, and tested for correctness across our experimental setups. Furthermore, all AI-polished text was carefully reviewed and edited by the authors to ensure scientific accuracy, structural coherence, and the preservation of the original meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our research focuses on improving the structural efficiency and mathematical tractability of generative models through the proposed plain architecture. While our method inherently advances the capabilities of generative modeling, it also shares the broader ethical considerations associated with this field, such as the potential generation of biased, harmful, or misleading content (e.g., deepfakes) if the models are maliciously deployed. However, our primary contribution is architectural and foundational in nature, aiming to democratize high-quality generative training by resolving computational bottlenecks. All experiments in this work were conducted using standard, publicly available datasets (e.g., ImageNet), and we adhere strictly to established data release practices. We are committed to responsible AI development and do not foresee any immediate or specific negative societal impacts directly arising from our architectural improvements.

REPRODUCIBILITY STATEMENT

We have made concerted efforts to ensure the full reproducibility of our proposed method and empirical findings. The detailed formulation of the plain flow maps, including the analytic bilinear head design and the corresponding trajectory parameterizations, is thoroughly documented in the main text. To facilitate the exact replication of our quantitative results on image generation benchmarks, we provide comprehensive details of the experimental configurations, including hyperparameter settings, network architectures, and dataset processing pipelines, in the Appendix. Furthermore, our complete source code—containing training scripts, inference pipelines, and pretrained network weights will be made publicly available upon publication.

REFERENCES

Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023a.

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 22563–22575, 2023b.

Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*, 38: 33346–33382, 2026.

Nicholas M Boffi, Michael S Albergo, and Eric Vanden-Eijnden. Flow map matching. *arXiv preprint arXiv:2406.07507*, 2024.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. pi-flow: Policy-based few-step generation via imitation distillation. In *International Conference on Learning Representations*, volume 2026, pp. 151521–151547, 2026.

Huayu Chen, Kai Jiang, Kaiwen Zheng, Jianfei Chen, Hang Su, and Jun Zhu. Visual generation without guidance. *arXiv preprint arXiv:2501.15420*, 2025a.

Jiuhai Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, et al. Blip3-o: A family of fully open unified multimodal models-architecture, training and dataset. *arXiv preprint arXiv:2505.09568*, 2025b.

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *International conference on learning representations*, volume 2024, pp. 57611–57640, 2024.

Junsong Chen, Shuchen Xue, Yuyang Zhao, Jincheng Yu, Sayak Paul, Junyu Chen, Han Cai, Song Han, and Enze Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16185–16195. IEEE, 2025c.

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

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J Zico Kolter. Consistency models made easy. *arXiv preprint arXiv:2406.14548*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30467–30476, 2026.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Joshua M Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2023.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion: 1.5 fid on imagenet512 with pixel-space diffusion. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18062–18071. IEEE, 2025.

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. Ella: Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024.

Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning*, pp. 13916–13932. PMLR, 2023.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10124–10134, 2023.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*, 2023.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. `https://bfl.ai/blog/flux-2`, 2025.

Kyungmin Lee, Sihyun Yu, and Jinwoo Shin. Decoupled meanflow: Turning flow models into flow maps for accelerated sampling. In *International Conference on Learning Representations*, volume 2026, pp. 64555–64583, 2026.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37: 56424–56445, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

Wenze Liu and Xiangyu Yue. Learning to integrate diffusion odes by averaging the derivatives. *arXiv preprint arXiv:2505.14502*, 2025.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Cheng Lu and Yang Song. Simplifying, stabilizing and scaling continuous-time consistency models. *arXiv preprint arXiv:2410.11081*, 2024.

Tianze Luo, Haotian Yuan, and Zhuang Liu. Soflow: Solution flow models for one-step generative modeling. In *International Conference on Learning Representations*, volume 2026, pp. 101623–101641, 2026.

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*, pp. 23–40. Springer, 2024.

Zehong Ma, Ruihan Xu, and Shiliang Zhang. Pixelgen: Pixel diffusion beats latent diffusion with perceptual loss. *arXiv preprint arXiv:2602.02493*, 2026.

Ziqi Pang, Tianyuan Zhang, Fujun Luan, Yunze Man, Hao Tan, Kai Zhang, William T Freeman, and Yu-Xiong Wang. Randar: Decoder-only autoregressive visual generation in random orders. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 45–55. IEEE, 2025.

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.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

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*, pp. 10684–10695, 2022.

Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. In *ACM SIGGRAPH 2022 conference proceedings*, pp. 1–10, 2022.

Axel Sauer, Frederic Boesel, Tim Dockhorn, Andreas Blattmann, Patrick Esser, and Robin Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024a.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. In *European Conference on Computer Vision*, pp. 87–103. Springer, 2024b.

Neta Shaul, Chao Liu, Arash Vahdat, and Julius Berner. Parallel decoding distillation for fast image and video generation. *arXiv preprint arXiv:2607.26004*, 2026.

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

Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. *arXiv preprint arXiv:2310.14189*, 2023.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Zhicong Tang, Jianmin Bao, Dong Chen, and Baining Guo. Diffusion models without classifier-free guidance. *arXiv preprint arXiv:2502.12154*, 2025.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. Ddt: Decoupled diffusion transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 40633–40642, 2026a.

Zidong Wang, Yiyuan Zhang, Xiaoyu Yue, Xiangyu Yue, Yangguang Li, Wanli Ouyang, and Lei Bai. Transition models: Rethinking the generative learning objective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 29178–29189, 2026b.

Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, et al. Omnigen2: Exploration to advanced multimodal generation. *arXiv e-prints*, pp. arXiv–2506, 2025.

Zihan Yang, Shuyuan Tu, Licheng Zhang, Qi Dai, Yu-Gang Jiang, and Zuxuan Wu. Arcflow: Unleashing 2-step text-to-image generation via high-precision non-linear flow distillation. *arXiv preprint arXiv:2602.09014*, 2026.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15703–15712, 2025.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Huijie Zhang, Aliaksandr Siarohin, Willi Menapace, Michael Vasilkovsky, Sergey Tulyakov, Qing Qu, and Ivan Skorokhodov. Alphaflow: Understanding and improving meanflow models. In *International Conference on Learning Representations*, volume 2026, pp. 144056–144094, 2026.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. *arXiv preprint arXiv:2510.11690*, 2025.

Linqi Zhou, Stefano Ermon, and Jiaming Song. Inductive moment matching. *arXiv preprint arXiv:2503.07565*, 2025a.

Linqi Zhou, Mathias Parger, Ayaan Haque, and Jiaming Song. Terminal velocity matching. *arXiv preprint arXiv:2511.19797*, 2025b.

APPENDIX

A EXPERIMENTAL SETUP FOR THE MOTIVATING STUDY

Our evaluation uses the official CFG=2.0 TVM-XL/2 checkpoint alongside a custom TVM-B/2 model. The TVM-B/2 model adopts the standard DiT-B/2 architecture (12 Transformer blocks, a hidden size of 768, and 12 attention heads) with CFG=2.0, and is trained for 300K iterations with a batch size of 1024. All other configurations strictly follow the original TVM settings.

For the evaluation of derivative contribution, we use 2048 ImageNet-256×256 training examples, noting that increasing this number to 4096 yields negligible differences in the results. For each sample, we draw a latent x_0, noise $\epsilon \sim \mathcal{N}(0, I)$, and a training-distributed time pair (t, s) following the original TVM sampling scheme.

Let $f(c_0, \ldots, c_D)$ denote the model output, where c_i is the terminal-time conditioning at site i. At injection index zero, all sites receive the same conditioning $c(s)$. Writing $\dot{c} = \partial_s c(s)$, the full terminal derivative and the contribution of site i are

$$J = \frac{\mathrm{d}}{\mathrm{d}s} f(c(s), \ldots, c(s)), \qquad J_i = \mathrm{JVP}_{c_i} f[\dot{c}]. \tag{23}$$

Each J_i is computed by applying the conditioning tangent only at site i, such that

$$J = \sum_{i=0}^{D} J_i. \tag{24}$$

For each site, we report its signed contribution

$$C_i = 100 \frac{\sum_n \langle J_{n,i}, J_n \rangle}{\sum_n |J_n|_2^2}, \tag{25}$$

and derivative strength

$$M_i = 100 \sqrt{\frac{\sum_n |J_{n,i}|_2^2}{\sum_n |J_n|_2^2}}. \tag{26}$$

The signed contributions retain directional cancellation and therefore sum to 100%, whereas the derivative strengths measure response magnitude and are not additive.

B ANALYTICAL DIFFERENTIATION AND INTEGRATION OF THE FLOW-MAP HEADS

Following the notation in the main text, we write $z_t = g_\phi(x_t, t)$ and $f_\theta(x_t, t, s) = h_\psi(z_t, s)$, with $\theta = (\phi, \psi)$. This section gives the explicit head derivatives and primitives required by the transition operators $\mathcal{T}_{t,s}^{\mathrm{sec}}$ and $\mathcal{T}_{t,s}^{\mathrm{vel}}$ to evaluate F_θ and its terminal derivative. We express the terminal-time dependence through the gap coordinate $\delta = t - s$. The source time t is held fixed in this coordinate change. Throughout, a prime denotes a derivative with respect to the gap, whereas ∂_s denotes a terminal-time derivative at fixed (x_t, t). Conditioning variables, the batch dimension, and the fixed token-to-latent reshaping are omitted for clarity.

B.1 FOURIER BASIS AND AFFINE TEMPORAL MODULATION

For the linear and bilinear heads, we use $2M$ fixed sinusoidal features, ordered as all cosine components followed by all sine components:

$$e(\delta) = [\cos(\omega_1 \delta), \ldots, \cos(\omega_M \delta), \sin(\omega_1 \delta), \ldots, \sin(\omega_M \delta)]^\top, \qquad \omega_m = m\omega_0, \quad \omega_0 = \frac{\lambda}{M}. \tag{27}$$

Our default is $2M = 256$ and $\lambda = 1$. Its derivative and definite primitive are

$$e'(\delta) = \left[\{-\omega_m \sin(\omega_m \delta)\}_{m=1}^{M},\ \{\omega_m \cos(\omega_m \delta)\}_{m=1}^{M}\right]^\top, \tag{28}$$

$$E(\delta) = \int_0^\delta e(q)\,\mathrm{d}q = \left[\{\sin(\omega_m \delta)/\omega_m\}_{m=1}^{M},\ \{[1 - \cos(\omega_m \delta)]/\omega_m\}_{m=1}^{M}\right]^\top. \tag{29}$$

For any affine modulation $m(\delta) = W_m e(\delta) + b_m$, therefore,

$$m'(\delta) = W_m e'(\delta), \qquad \overline{m}(\delta) := \int_0^\delta m(q)\,\mathrm{d}q = W_m E(\delta) + \delta b_m. \tag{30}$$

In particular, a projection bias has zero derivative but contributes δb_m to the integral. Embedding, differentiation, and integration all use the same frequency buffer and learned projection parameters. Converting these expressions to the terminal-time coordinate introduces a minus sign:

$$\partial_s m(t-s) = -m'(\delta), \qquad \int_t^s m(t-r)\,\mathrm{d}r = -\overline{m}(\delta). \tag{31}$$

B.2 Linear AdaLN head

The linear head provides a simple reference case. Define $\widetilde{z}_t = \mathrm{RMSNorm}(z_t)$ and let $a(\delta)$ and $b(\delta)$ be affine modulations of the form in Eq. equation 30. With a shared, bias-free output projection $\mathcal{P}$, the head is

$$h_\psi(z_t, s) = \mathcal{P}[\widetilde{z}_t \odot (1 + a(\delta)) + b(\delta)]\,. \tag{32}$$

Channel-wise modulations are broadcast over the token dimension. Since $\widetilde{z}_t$ is independent of the terminal time,

$$\partial_s h_\psi(z_t, s) = -\mathcal{P}[\widetilde{z}_t \odot a'(\delta) + b'(\delta)]\,, \tag{33}$$

$$\int_t^s h_\psi(z_t, r)\,\mathrm{d}r = -\mathcal{P}\big[\widetilde{z}_t \odot \big(\delta + \overline{a}(\delta)\big) + \overline{b}(\delta)\big]\,. \tag{34}$$

These are respectively $\partial_s f_\theta(x_t, t, s)$ and $\mathcal{T}^{\mathrm{vel}}_{t,s}[f_\theta(x_t, t, \cdot)]$ in the notation of the main text. The integral vanishes at $\delta = 0$ without requiring a separate displacement head.

B.3 Bilinear attention head

The bilinear head adds terminal-time-dependent token interactions. For one attention head of width d_h, source-only projections of $\mathrm{RMSNorm}(z_t)$ produce $Q, K, V \in \mathbb{R}^{N \times d_h}$. The temporal key and value scales, $k(\delta)$ and $v(\delta)$, are affine functions of $e(\delta)$. Define

$$K_\delta = K \odot k(\delta), \qquad V_\delta = V \odot v(\delta), \tag{35}$$

$$A(\delta) = \kappa Q(K_\delta^\top V_\delta), \qquad \kappa = (N d_h)^{-1/2}. \tag{36}$$

Attention outputs are concatenated across heads. For brevity, A below denotes the resulting full-width token tensor. The output is

$$h_\psi(z_t, s) = \mathcal{P}\big[\big(z_t + A(\delta)\big) \odot (1 + a(\delta)) + b(\delta)\big]\,. \tag{37}$$

The residual is the original z_t, not its normalized version. All normalization is performed before temporal modulation; neither softmax nor time-dependent normalization is applied to the attention output.

Analytical differentiation. The product rule gives the gap derivative, separately for each attention head,

$$A'(\delta) = \kappa Q\left[(K \odot k'(\delta))^\top V_\delta + K_\delta^\top (V \odot v'(\delta))\right]. \tag{38}$$

After concatenation and conversion to the terminal-time coordinate, the complete head derivative is

$$\partial_s h_\psi(z_t, s) = -\mathcal{P}[A'(\delta) \odot (1 + a(\delta)) + (z_t + A(\delta)) \odot a'(\delta) + b'(\delta)]\,. \tag{39}$$

This expression includes the contribution $-\mathcal{P}[z_t \odot a'(\delta)]$ from the output modulation on the residual branch.

Analytical integration. Let $C = K^\top V$. Within one attention head, the temporally modulated context has entries $C_{ij}k_i(q)v_j(q)$. Including output modulation requires the matrix-valued primitive

$$T_{ij}(\delta) = \int_0^\delta k_i(q)v_j(q)[1 + a_j(q)]\,\mathrm{d}q. \tag{40}$$

Let

$$J(\delta) = \mathrm{Concat}_h\left[\kappa_h Q_h\big(C_h \odot T_h(\delta)\big)\right]. \tag{41}$$

Linearity of the bias-free projection and the change of variable $q = t - r$ then yield

$$\int_t^s h_\psi(z_t, r)\,\mathrm{d}r = -\mathcal{P}\big[J(\delta) + z_t \odot \big(\delta + \overline{a}(\delta)\big) + \overline{b}(\delta)\big]. \tag{42}$$

Crucially, T integrates the *product* of the three temporal factors, not the product of their individual integrals.

Exact finite Fourier representation. Each factor in Eq. equation 40 has harmonic degree at most M, including a possible constant term. Their product therefore has degree at most $L = 3M$:

$$k_i(q)v_j(q)[1 + a_j(q)] = c_{ij,0} + \sum_{n=1}^{L}\left[c_{ij,n}\cos(n\omega_0 q) + d_{ij,n}\sin(n\omega_0 q)\right]. \tag{43}$$

Thus,

$$T_{ij}(\delta) = c_{ij,0}\delta + \sum_{n=1}^{L}\left[c_{ij,n}\frac{\sin(n\omega_0\delta)}{n\omega_0} + d_{ij,n}\frac{1 - \cos(n\omega_0\delta)}{n\omega_0}\right]. \tag{44}$$

The zero-frequency component is integrated as $c_{ij,0}\delta$, avoiding division by zero when frequency combinations cancel.

The implementation evaluates this primitive without explicitly forming all Fourier coefficients. Let $S = 2L + 1$, $\varphi_\ell = 2\pi\ell/S$, and $q_\ell = \varphi_\ell/\omega_0$, for $\ell = 0, \ldots, S - 1$. Define

$$\rho_\ell(\delta) = \frac{\delta}{S} + \frac{2}{S}\sum_{n=1}^{L}\left[\frac{\sin(n\omega_0\delta)}{n\omega_0}\cos(n\varphi_\ell) + \frac{1 - \cos(n\omega_0\delta)}{n\omega_0}\sin(n\varphi_\ell)\right]. \tag{45}$$

Discrete Fourier orthogonality gives the equivalent expression

$$T_{ij}(\delta) = \sum_{\ell=0}^{S-1}\rho_\ell(\delta)k_i(q_\ell)v_j(q_\ell)[1 + a_j(q_\ell)]. \tag{46}$$

Because $S = 2L+1$ resolves every harmonic in the product, this is an exact finite Fourier identity in real arithmetic, not an approximate time-stepping or numerical quadrature rule. Only the lightweight temporal modulations are evaluated on this fixed phase grid; the backbone is not reevaluated. With our default $M = 128$, the grid has $S = 769$ points.

B.4 Nonlinear DiT head: differentiation only

For the DiT-head ablation, we retain the standard DiT timestep MLP and final AdaLN layer, with the terminal gap as the timestep input. This head uses the usual logarithmically spaced timestep frequencies, $\nu_m = \lambda\exp[-\log(10000)m/M]$ for $m = 0, \ldots, M - 1$, rather than the uniform harmonics in Eq. equation 27. Denote its sinusoidal features by $e_{\mathrm{D}}(\delta)$ and write

$$\eta = W_1 e_{\mathrm{D}}(\delta) + b_1, \qquad \xi = W_2\sigma(\eta) + b_2, \tag{47}$$

$$\begin{bmatrix} b(\delta) \\ a(\delta) \end{bmatrix} = W_3\sigma(\xi) + b_3, \qquad \widetilde{z}_t = \mathrm{LayerNorm}(z_t), \tag{48}$$

where $\sigma(x) = x\,\mathrm{sigmoid}(x)$ is SiLU. The head output is

$$h_\psi(z_t, s) = \mathcal{W}[\widetilde{z}_t \odot (1 + a(\delta)) + b(\delta)] + b_{\mathrm{out}}, \tag{49}$$

with $\mathcal{W}$ denoting the linear part of the output projection. Using

$$\sigma'(x) = \mathrm{sigmoid}(x) + x\,\mathrm{sigmoid}(x)[1 - \mathrm{sigmoid}(x)], \tag{50}$$

the modulation gap derivatives and the head terminal-time derivative follow directly from the chain rule:

$$\eta' = W_1 e'_{\mathrm{D}}(\delta), \tag{51}$$

$$\xi' = W_2[\sigma'(\eta) \odot \eta'], \tag{52}$$

$$\begin{bmatrix} b'(\delta) \\ a'(\delta) \end{bmatrix} = W_3[\sigma'(\xi) \odot \xi'], \tag{53}$$

$$\partial_s h_\psi(z_t, s) = -\mathcal{W}[\widetilde{z}_t \odot a'(\delta) + b'(\delta)]\,. \tag{54}$$

The output bias is absent from the derivative but remains in h_ψ and hence in the secant displacement and terminal velocity. The nonlinear temporal MLP does not preserve the finite Fourier family, so this head is supported only by Plain-Secant; no closed-form integral is used for it.

B.5 Gradient computation and numerical implementation

The equations above replace automatic differentiation with respect to terminal time by explicit tensor expressions. Training still uses ordinary reverse-mode backpropagation through these expressions to update both the head parameters ψ and the backbone parameters ϕ. No backbone JVP, terminal-time autograd derivative, finite-difference derivative, or numerical ODE solver is required to construct the flow map and its terminal velocity. Any stop-gradient in the target branch of the training objective is separate from this parameterization.

The identities hold in real arithmetic; their implementations are subject to floating-point error. In our experiments, BF16 mixed precision preserves sufficient accuracy for the explicit differentiation in Plain-Secant, but can produce substantial numerical errors in the bilinear Plain-Velocity integral. In the bilinear head, temporal modulation involves the product of three Fourier-valued factors: the key modulation, value modulation, and output modulation. Its closed-form integral combines contributions that can substantially cancel, making the result sensitive to BF16 rounding. To address this, we disable autocasting and evaluate the complete bilinear Plain-Velocity integral in FP32, including the source Q/K/V projections, temporal products, context contractions, and output projection. Plain-Secant retains the configured mixed-precision head operations, with trigonometric time features evaluated in FP32.

C Implementation Details

C.1 Analytic Head Implementation

Table 5: Configurations of the flow-map heads.

Configuration	Linear head	Bilinear head	DiT head
time cond	$\delta = t - s$	$\delta = t - s$	$\delta = t - s$
hidden dim	same as backbone	same as backbone	same as backbone
time embedding dim	256	256	256
frequency spacing	Uniform	Uniform	Logarithmic
time scale	1.0	1.0	1.0
time embedding MLP	None	None	Linear–SiLU–Linear
source-feature norm	RMSNorm	RMSNorm	LayerNorm
attention layers	0	1	0
attention heads	-	same as backbone	-
output projection bias	✗	✗	✓
supported parameterizations	secant / velocity	secant / velocity	secant

Table 5 details the configurations for the linear, bilinear, and DiT heads. For both the linear and bilinear heads, the RMSNorm scale is initialized to 1. In the bilinear head, the query, key, and value projection weights are initialized randomly with zero biases. Furthermore, its key and value

modulation weights are initialized to 0, with their corresponding biases set to 1. Across all three architectures, the output scale and shift projection weights, as well as their biases, are initialized to 0. All final output projections, including any present output biases, are strictly zero-initialized. Lastly, for the DiT head, the time-embedding MLP weights are randomly initialized with zero biases.

C.2 IMAGENET CLASS-TO-IMAGE GENERATION

Data space and model architecture. We utilize the standard pretrained SD-VAE[1] and train generative models in the latent space. We retain the original DiT (Peebles & Xie, 2023) architecture and replace its head with our analytic head.

Adaptive loss normalization. Following established practices (Song & Dhariwal, 2023; Geng et al., 2024; 2025), we incorporate an adaptive normalization into the terminal loss. We assign inverse-error weights of $1/\,\mathrm{BN}(\|\Delta\|^2+\epsilon)^p$, where Δ denotes the terminal-velocity residual. Here, BN normalizes the values to a unit mean within the batch, ensuring that the relative scale between the flow-matching and terminal objectives remains unchanged. We set $p=1$ as in previous works (Geng et al., 2025; Luo et al., 2026). To ensure the original learning dynamics remain intact, we set a large parameter $\epsilon = 0.1$, whereas the typical terminal loss magnitude is only around 0.005. Consequently, our objective operates essentially as a standard MSE loss, intervening solely to suppress extreme numerical outliers. This sharply contrasts with prior methods that employ a small ϵ, which fundamentally alters the loss formulation away from exact MSE. Empirically, our training process is fully stable without this module, and its inclusion does not affect general generative performance. Instead, it serves as a safety lock.

Classifier-free guidance. To condition the network on the classifier-free guidance (CFG) scale w, we adopt the sampling strategy from Geng et al. (2026):

$$p(w) \propto w^{-\beta_{\mathrm{CFG}}}, \qquad 1 \le w \le 6, \tag{55}$$

where β_{CFG} controls the skew intensity. We encode the term $(1-1/w)$ using a sinusoidal embedding followed by a two-layer MLP, adding the resulting features to the timestep and class embeddings. The guidance interval remains fixed to $[0, 0.7]$. The CFG scales used in evaluation are illustrated in Table 6, while the CFG scale used for visualization is set to 5.

Table 6: Guidance scales used for ImageNet evaluation.

Parameterization	Head	NFE	B/2	M/2	L/2	XL/2
Plain-Secant	Bilinear	1	3.6	3.2	2.8	2.9
Plain-Secant	Bilinear	2	-	-	-	2.5
Plain-Velocity	Bilinear	1	3.7	3.2	3.0	2.9
Plain-Velocity	Bilinear	2	-	-	-	2.6
Plain-Secant	Linear	1	2.9	-	-	-
Plain-Velocity	Linear	1	2.6	-	-	-
Plain-Secant	DiT	1	3.3	2.9	2.6	2.6

Training efficiency comparison. Following the benchmarking protocol of TVM (Zhou et al., 2025b), we evaluate training efficiency on a single node equipped with 8 NVIDIA H100 GPUs using bfloat16 mixed precision. We disable the model EMA and omit advanced compilation optimizations such as `torch.compile`. Under this setup, we report the average per-step runtime alongside the peak memory consumption per GPU.

We summarize the training hyperparameters and experimental configurations in Table 7.

C.3 TEXT-TO-IMAGE GENERATION

We execute text-to-image generation within the latent space of the preatrained FLUX.2 VAE (Labs, 2025). Following PixelGen (Ma et al., 2026), a trainable text refiner of depth 4 processes the text features prior to the transformer. The training pipeline comprises initial pretraining at 256×256

[1] https://huggingface.co/stabilityai/sd-vae-ft-mse

Table 7: Model and training configurations of plain flow map on ImageNet-256 $\times$ 256. Settings apply to both Plain-Secant and Plain-Velocity with the bilinear analytic head.

Configuration	B/2	M/2	L/2	XL/2
parameters (M)	133	310	462	679
image resolution	256×256			
latent shape	$32 \times 32 \times 4$			
patch size	2×2			
depth	12	16	24	28
hidden dim	768	1024	1024	1152
heads	12	16	16	16
training steps	300K			
batch size	2048			
optimizer	AdamW			
learning rate	1×10^{-4}			
Adam (β_1, β_2)	$(0.9, 0.99)$			
weight decay	0			
EMA decay	0.9999			
loss-target EMA	0.99			
precision	BF16 mixed			
dropout	0.0			
time cond	backbone: source time t; head: gap $\delta = t - s$ only			
time sampling	Gap $(-1.6, 1.0), (-0.4, 1.0)$ (Zhou et al., 2025b)			
CFG class drop	0.1			
CFG exponent β_{CFG}	1.0	2.0	2.0	2.0
CFG active interval	$[0, 0.7]$			

resolution, continued pretraining at 512×512, and supervised fine-tuning at 512×512. The pretraining stages incorporate a mixture of the JourneyDB, long-caption, and short-caption subsets from BLIP3o (Chen et al., 2025b), whereas the final fine-tuning phase utilizes BLIP3o-60K (Chen et al., 2025b). We also incorporate adaptive loss normalization with $p = 1$ and $\epsilon = 0.1$. We apply the identical CFG-scale embedding and sampling scheme used in our ImageNet experiments, and enable guidance across the full time interval $[0, 1]$. Further configurations are detailed in Table 8.

D Additional Results and Metrics

In this section, we provide supplementary quantitative results.

Effect of the CFG scale on linear and bilinear heads. Figure 7 illustrates the FID curves for both the linear and bilinear heads across various CFG values. This comparison demonstrates that the optimal CFG scale differs depending on the chosen parameterization.

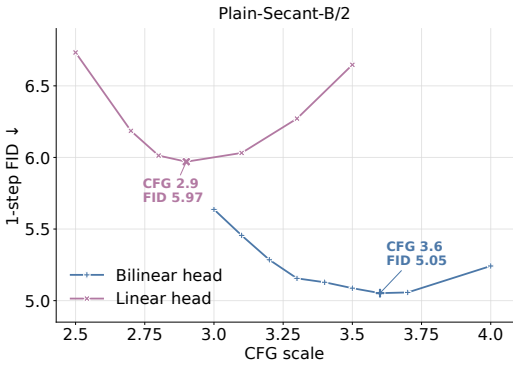


Figure 7: Impact of the CFG scale on the linear and bilinear heads.

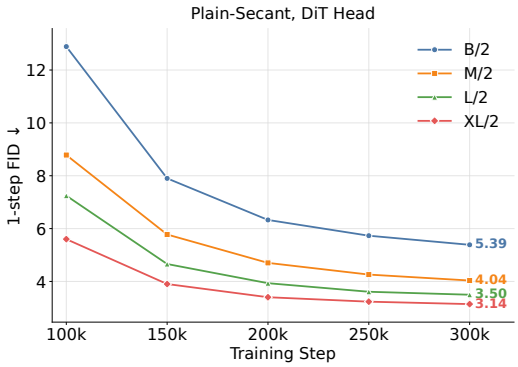


Figure 8: Scaling behavior of Plain-Secant equipped with a DiT head.

Table 8: Model and training configurations of Plain-Secant for text-to-image generation.

Configuration	Pretrain-256	Pretrain-512	Finetune-512
parameters (M)		1132	
image resolution	256×256	512×512	512×512
latent shape	$32 \times 32 \times 32$	$64 \times 64 \times 32$	$64 \times 64 \times 32$
patch size		2×2	
depth		16	
hidden dim		1536	
heads		24	
text encoder		Qwen3-1.7B	
text length		128	
text dim		2048	
training steps	200K	80K	60K
batch size	1536	512	512
optimizer		AdamW	
learning rate	2×10^{-4}	1×10^{-4}	1×10^{-5}
Adam (β_1, β_2)		$(0.9, 0.95)$	
weight decay		0	
EMA decay		0.9999	
loss-target EMA	0.99	0.99	0.9
precision		BF16 mixed	
dropout		0.0	
time cond	backbone: source time t; head: gap $\delta = t - s$ only		
time sampling	Gap $(-1.6, 1.0), (-0.4, 1.0)$ (Zhou et al., 2025b)		
CFG caption drop		0.1	
CFG exponent β_{CFG}		2.0	
CFG active interval		$[0, 1]$	

Scaling behavior of Plain-Secant with DiT head. The results provided in Figure 8 confirm that Plain-Secant with DiT head also scales predictably and efficiently as the parameter count grows.

Extended system-level comparisons on ImageNet. We provide a broader system-level comparison on ImageNet-256×256 in Table 9.

Full metrics on GenEval and DPG-Bench. Due to space constraints, the main manuscript reports only the overall scores for our text-to-image generation experiments. In Table 10 and Table 11, we present the complete evaluation metrics for both the GenEval and DPG-Bench benchmarks, respectively.

E Additional Visualizations

In this section, we provide extended qualitative results for both ImageNet class-conditional generation and text-to-image synthesis.

For the ImageNet benchmark, we present additional uncurated samples. The 1-NFE generation results are displayed in Figure 9 and Figure 10. Similarly, the corresponding 2-NFE samples are visualized in Figure 11 and Figure 12.

Furthermore, we include supplementary text-to-image generation results evaluated at a 4-NFE budget in Figure 13. The corresponding textual prompts are provided below, ordered from left to right and top to bottom.

(1) A tiny house sheltered beneath a giant leaf.

(2) A paper-cut whale above indigo waves.

(3) An ink painting of a lonely cliffside pine.

(4) An alpine shepherd knitting beside a sheep.

(5) A portrait of an industrial chef wearing a white ceramic toque.

Table 9: Class-conditional generation on ImageNet 256×256. † reported in Zhou et al. (2025a).

Method	Params	NFE	FID ↓
GANs			
BigGAN (Brock et al., 2018)	112M	1	6.95
GigaGAN (Kang et al., 2023)	569M	1	3.45
StyleGAN-XL (Sauer et al., 2022)	166M	1	2.30
masked and autoregressive models			
MaskGIT (Chang et al., 2022)	227M	8	6.18
LlamaGen-XL (Sun et al., 2024)	775M	576	2.62
VAR (Tian et al., 2024)	2.0B	10	1.80
MAR (Li et al., 2024)	943M	64	1.55
RandAR-XL (Pang et al., 2025)	775M	256	2.22
diffusion models			
ADM-G (Dhariwal & Nichol, 2021)	554M	250×2	4.59
LDM-4-G (Rombach et al., 2022)	395M	250×2	3.60
DiT-XL/2 (Peebles & Xie, 2023)	675M	250×2	2.27
SiT-XL/2 (Ma et al., 2024)	675M	250×2	2.06
SiT-REPA-XL/2 (Yu et al., 2024)	675M	250×2	1.42
SiD2 (Hoogeboom et al., 2025)	-	512×2	1.38
LightningDiT-XL/2 (Yao et al., 2025)	675M	250×2	1.35
DDT-XL/2 (Wang et al., 2026a)	675M	250×2	1.26
RAE-DiT^{DH}-XL (Zheng et al., 2025)	839M	250×2	1.13
few-step diffusion models from scratch			
iCT-XL/2 (Song & Dhariwal, 2023)	675M	1 / 2	34.24 / 20.30
Shortcut-XL/2 (Frans et al., 2024)	675M	1 / 4	10.60 / 7.80
IMM-XL/2 (Zhou et al., 2025a)	675M	1×2 / 2×2	7.77 / 3.99
MeanFlow-XL/2 (Geng et al., 2025)	676M	1 / 2	3.43 / 2.93
TiM-XL/2 (Wang et al., 2026b)	664M	1	3.26
α-Flow-XL/2 (Zhang et al., 2026)	676M	1 / 2	2.81 / 2.16
iMF-XL/2 (Geng et al., 2026)	610M	1 / 2	1.72 / 1.54
SoFlow-XL/2 (Luo et al., 2026)	676M	1 / 2	2.96 / 2.66
TVM-XL/2 (Zhou et al., 2025b)	678M	1 / 2	3.29 / 2.80
Plain-Secant-XL/2 (Ours)	679M	1 / 2	2.89 / 2.40
Plain-Velocity-XL/2 (Ours)	679M	1 / 2	2.92 / 2.43

Table 10: Full evaluation metrics for text-to-image generation on the GenEval benchmark.

Method	Params	NFE	Sin.Obj.	Two.Obj.	Counting	Colors	Pos.	Color Attr.	Overall ↑
Diffusion models									
PixArt-α (Chen et al., 2024)	0.6B	20 × 2	0.98	0.50	0.44	0.80	0.08	0.07	0.48
SD3-8B (Esser et al., 2024)	8B	50 × 2	0.98	0.84	0.66	0.74	0.40	0.43	0.68
FLUX.1-dev (Labs, 2024)	12B	50	0.99	0.81	0.79	0.74	0.20	0.47	0.67
DALL-E 3 (Betker et al., 2023)	-	-	0.96	0.87	0.47	0.83	0.43	0.45	0.67
OmniGen2 (Wu et al., 2025)	4B	50 × 2	1.00	0.95	0.64	0.88	0.55	0.76	0.80
PixelGen (Ma et al., 2026)	1.1B	25 × 2	0.99	0.88	0.59	0.90	0.70	0.70	0.79
Distilled few-step diffusion models									
SDXL-Turbo (Sauer et al., 2024b)	2.6B	8	0.99	0.75	0.07	0.89	0.11	0.20	0.50
SANA-Sprint (Chen et al., 2025c)	1.6B	8	1.00	0.88	0.56	0.87	0.56	0.47	0.72
SD3.5-Turbo (Sauer et al., 2024a)	8B	8	0.99	0.81	0.62	0.79	0.25	0.48	0.66
FLUX.1-Schnell	12B	1	0.99	0.88	0.63	0.78	0.27	0.53	0.68
From-scratch few-step diffusion models									
TiM (Wang et al., 2026b)	0.9B	1	0.98	0.75	0.52	0.80	0.54	0.44	0.67
Plain-Secant (Ours)	1.1B	1	0.97	0.88	0.64	0.85	0.71	0.63	0.78
		2	0.98	0.90	0.64	0.89	0.75	0.66	0.80
		4	1.00	0.90	0.64	0.93	0.79	0.70	0.83
		8	1.00	0.90	0.68	0.94	0.79	0.73	0.84

(6) A portrait of a scientist with translucent sky blue hair.

(7) A portrait of a dreamer beside floating porcelain moons.

(8) A floating cyberpunk temple beneath a radiant turquoise sun.

Table 11: Full evaluation metrics for text-to-image generation on DPG-Bench.

Method	Params	NFE	Global	Entity	Attribute	Relation	Other	Overall ↑
Diffusion models								
PixArt-α (Chen et al., 2024)	0.6B	20×2	74.97	79.32	78.60	82.57	76.96	71.11
SD3-8B (Esser et al., 2024)	8B	50×2	-	-	-	-	-	-
FLUX.1-dev (Labs, 2024)	12B	50	81.46	90.02	87.50	92.72	82.39	83.32
DALL-E 3 (Betker et al., 2023)	-	-	90.97	89.61	88.39	90.58	89.83	83.50
OmniGen2 (Wu et al., 2025)	4B	50×2	88.81	88.83	90.18	89.37	90.27	83.57
PixelGen (Ma et al., 2026)	1.1B	25×2	85.61	86.84	89.35	85.69	87.85	80.01
Distilled few-step diffusion models								
SDXL-Turbo (Sauer et al., 2024b)	2.6B	8	-	-	-	-	-	-
SANA-Sprint (Chen et al., 2025c)	1.6B	8	83.84	88.54	88.50	87.40	86.41	81.08
SD3.5-Turbo (Sauer et al., 2024a)	8B	8	80.12	86.13	84.73	91.86	78.29	79.03
FLUX.1-Schnell	12B	1	86.62	90.82	88.35	93.45	82.00	84.94
From-scratch few-step diffusion models								
TiM (Wang et al., 2026b)	0.9B	1	82.98	83.64	83.54	91.99	63.20	74.93
Plain-Secant (Ours)	1.1B	1	83.89	84.86	83.58	90.68	71.60	76.08
		2	83.28	85.96	84.28	91.26	72.00	77.65
		4	82.07	86.87	85.48	91.26	76.40	78.85
		8	79.33	87.92	86.09	92.07	77.20	80.11

(9) 青龙武士穿着淡翡翠色机甲。(An Azure Dragon warrior wearing pale emerald mecha.)

(10) 冰雪国王站在阳光下的洁白雪原。(The Ice King stands on a sunlit, pure white snowfield.)

(11) A porcelain owl unfolding intricate mechanical feathers.

(12) A cloud elephant with lightning crackling between its tusks.

(13) A girl wearing an aquarium helmet filled with fish.

(14) A porcelain rocket launching through cobalt blue clouds.

(15) A desert highway passing beneath an enormous Saturn.

(16) A low poly chrome fox in a neon landscape.

(17) A pixel art knight beneath a striped magenta moon.

(18) 发光白狼穿越结满冰晶的森林。(A glowing white wolf walking through an ice crystal forest.)

(19) 白瓷茶杯里翻涌着蓝色雷暴。(A blue thunderstorm surging in a white porcelain teacup.)

(20) An animated cheetah racer grins through an open helmet, orange ears swept back.

(21) A coral lion resting on an underwater throne.

(22) A felt rabbit inside a miniature kitchen.

(23) A striped hot-air balloon landing gently in a sunflower maze.

(24) A goldfish swimming inside a bubble above the city.

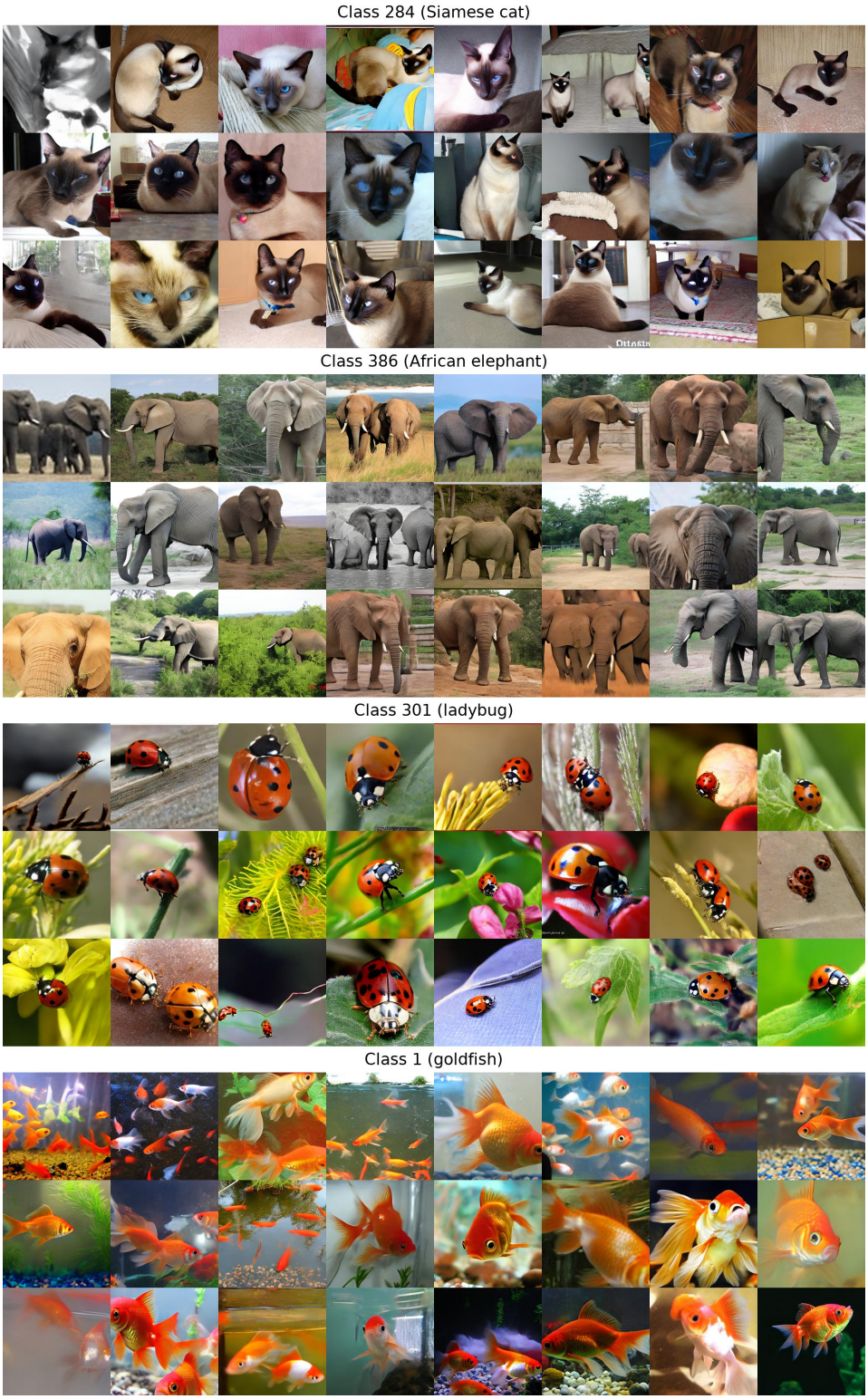


Figure 9: Uncurated 1-NFE samples generated by Plain-Secant-XL/2 (Part 1).

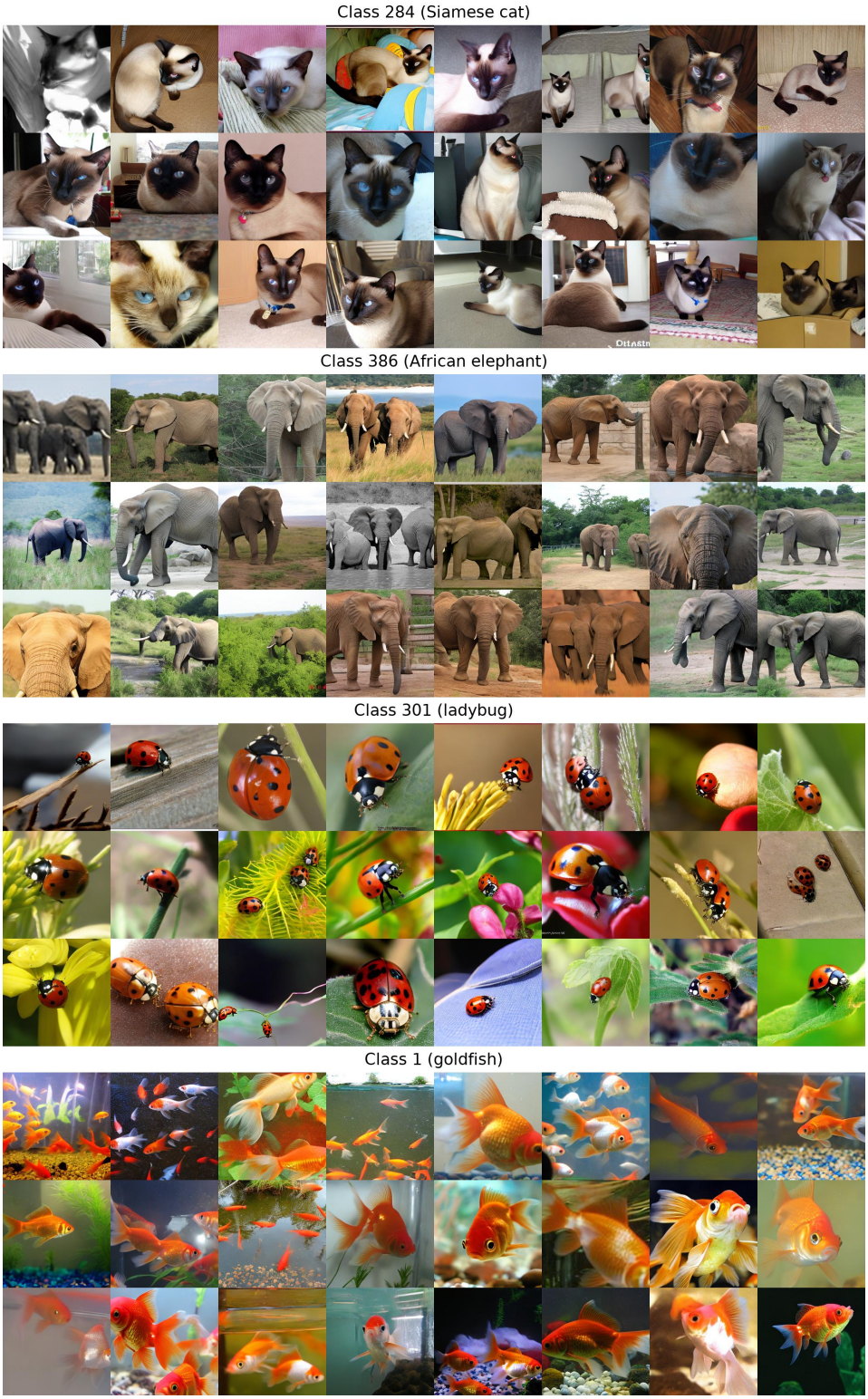


Figure 10: Uncurated 2-NFE samples generated by Plain-Secant-XL/2 (Part 1).

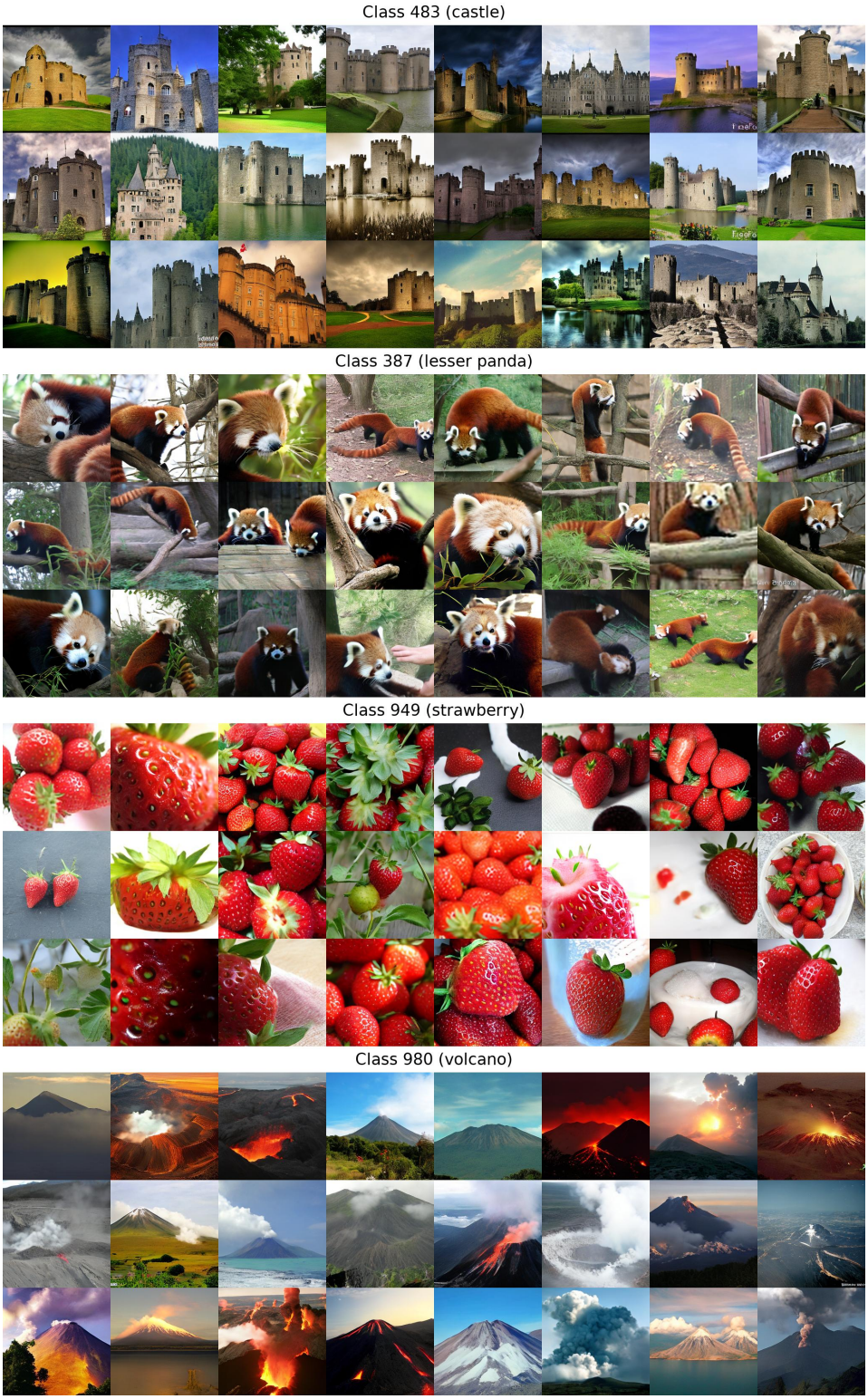


Figure 11: Uncurated 1-NFE samples generated by Plain-Secant-XL/2 (Part 2).

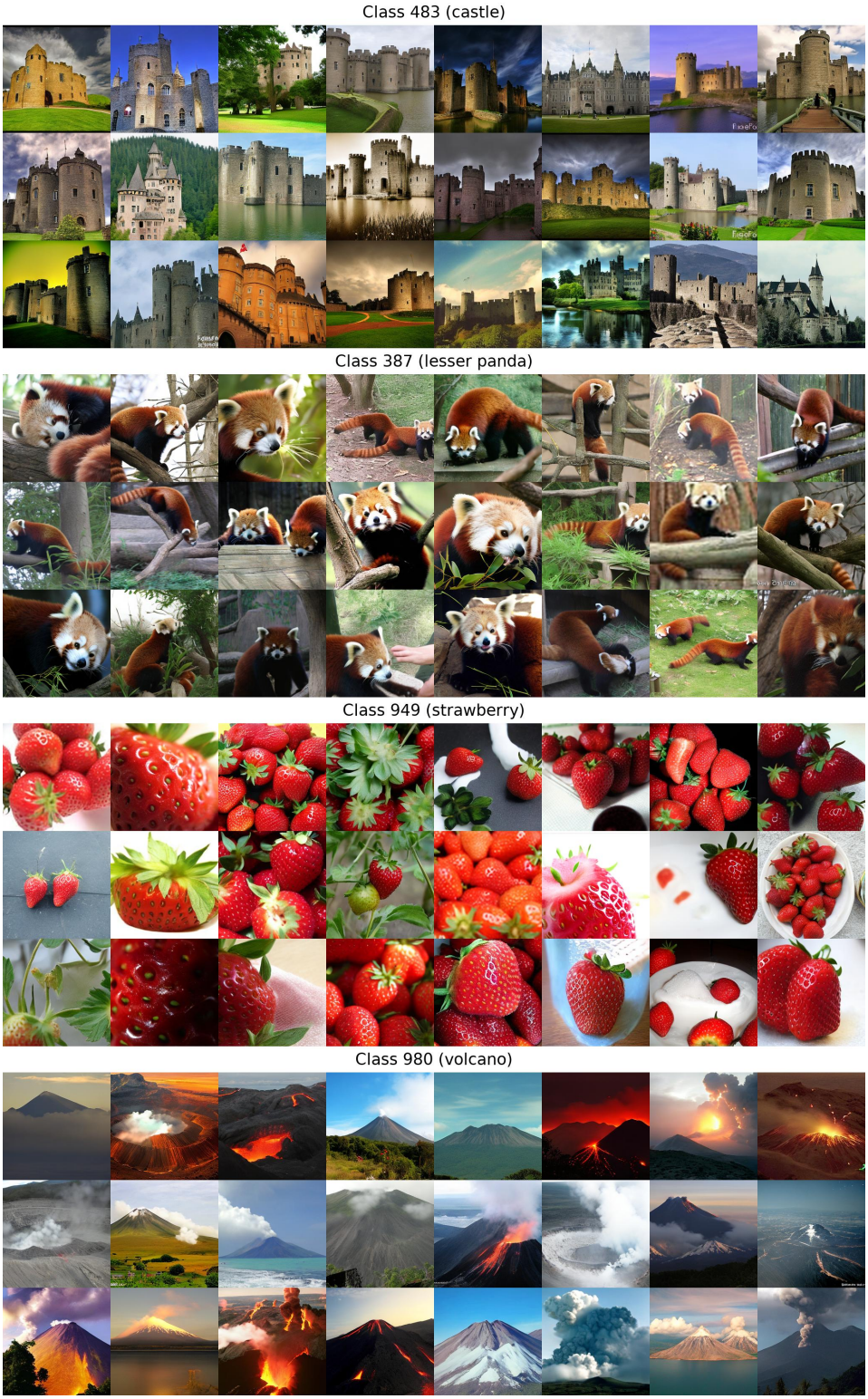


Figure 12: Uncurated 2-NFE samples generated by Plain-Secant-XL/2 (Part 2).



Figure 13: Additional qualitative results for 4-NFE text-to-image generation.