Decompose the Distillation: Interpretable Single-Pass Guidance for Diffusion Models

Wuyang Zhang **Xiaolin Xu***
Department of Electrical and Computer Engineering
Northeastern University, Boston, MA, USA
doggo@ieee.org, x.xu@northeastern.edu

Abstract

Classifier-Free Guidance (CFG) is a widely used inference-time guidance rule for conditional image generation with diffusion models, which exposes a guidance scale that adjusts conditioning strength after training. This flexibility requires conditional and unconditional model evaluations at every sampling step, so an N-step trajectory takes $2N$ forward passes. Single-pass alternatives reduce such cost, yet most collapse CFG's two components into one opaque guided prediction, losing the separable base flow and guidance direction that support per-step control and diagnostics, even when the scale remains adjustable. This work introduces CtrlFlow, a single-pass guidance method that preserves CFG's separable base-flow and guidance-direction interface. To preserve both components at one-pass cost, CtrlFlow attaches two lightweight LoRA adapters to a frozen backbone, each adding 2.5% parameters, and predicts the base flow and guidance direction from one pass of the shared frozen backbone. Because the guidance scale multiplies the guidance direction only after the forward pass, one evaluation yields the guided velocity at any scale, so a single trained checkpoint remains adjustable at inference without retraining. Preserving two outputs introduces a training mismatch: inference amplifies guidance-direction errors by the chosen scale, while naive component-wise losses weight both outputs equally. To align training with this composed inference-time error, CtrlFlow uses a unified multi-scale distillation loss. Residual guidance errors can still persist at extreme scales, where even small guidance-direction errors are amplified by the guidance scale. To reduce this inference-time error, CtrlFlow applies scale-adaptive modulation to damp velocity-field errors and time-grid warping to concentrate solver evaluations where the trajectory changes most rapidly. On ImageNet 256×256 with the DiT-XL/2 backbone, CtrlFlow matches CFG within a pre-specified tolerance in 24 of 25 (N, s) configurations, against 20 of 25 for a matched scale-conditioned opaque student, and achieves a measured $1.93\times$ wall-clock speedup at $N{=}25$.

1 Introduction

In conditional image generation with diffusion models [Ho et al., 2020, Song et al., 2021, Rombach et al., 2022], each sample should both look like a plausible natural image and match the conditioning signal, such as the requested class in class-conditional generation [Dhariwal and Nichol, 2021, Peebles and Xie, 2023]. Achieving both goals simultaneously is challenging: stronger guidance improves condition matching, whereas overly strong guidance can introduce visible artifacts [Sadat et al., 2025a] and make the model produce similar outputs instead of diverse valid images for the same condition [Karras et al., 2024, Kynkäänniemi et al., 2024]. Inference-time guidance methods [Dhariwal and Nichol, 2021, Ho and Salimans, 2021] address this trade-off by letting

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

practitioners control the strength of the conditioning signal during sampling. Classifier-Free Guidance (CFG) [Ho and Salimans, 2021], the most widely adopted such method and a foundation for many later variants, implements this control with a scalar guidance scale s that can be tuned after training. Because CFG compares two predictions from the same network at every sampling step, practitioners obtain per-sample, per-step control over the quality-alignment trade-off without retraining. This flexibility carries a cost: every guided step requires two network evaluations rather than one, so an N-step trajectory takes $2N$ forward passes and wall-clock latency nearly doubles [Ho et al., 2020, Song et al., 2021, Rombach et al., 2022].

Several recent works attacked this $2N$ overhead. Across these attempts, speed is typically gained by giving up some facet of CFG's interface, leaving practitioners to choose between latency and control. One route distills guidance into a single-pass student that takes the scale as a network input, as in AGD [Jensen and Sadat, 2025], which trains an adapter on a frozen backbone, and Guidance-Free Training (GFT) [Chen et al., 2025a], which fine-tunes the whole model. Both halve the forward passes and keep the scale adjustable, but each pass commits to one scale and returns an opaque guided velocity with no separable base flow or guidance direction. Model Guidance [Tang et al., 2025] and Entropy Rectifying Guidance [Ifriqi et al., 2025] take a different route, building guidance directly into the model. Both methods run in a single pass, yet both collapse the base/guidance split and surrender the post-hoc scale knob that motivated CFG. A complementary pattern keeps two passes and improves the guidance step instead. CFG-Zero* [Fan et al., 2025] and APG [Sadat et al., 2025a] sharpen the guided update, while retaining the conditional and unconditional evaluations. Autoguidance [Karras et al., 2024] makes the second evaluation cheaper by replacing the unconditional model with a smaller trained network. None of these two-pass refinements reduces the $2N$ evaluation count. For the goal of CFG-like control at lower latency, these trade-offs reduce to three axes. *Single-pass compute* measures whether each step uses one network evaluation or two. The criterion of *explicit access to* $\mathbf{v}_0$ *and* $\mathbf{g}$ concerns whether the method's inference interface keeps base flow and guidance direction separately available for control or diagnosis. The remaining axis, *post-hoc* s, captures whether the method exposes a guidance-scale knob that is not tied to a single training scale and can be varied at inference without retraining. Table 1 organizes existing methods along these axes. Scale-conditioned students such as AGD and GFT already combine single-pass compute with post-hoc s, so the missing piece is the interface: to our knowledge, no prior single-pass method keeps $\mathbf{v}_0$ and $\mathbf{g}$ separately available.

Table 1: Comparison of guidance methods. *Explicit* $\mathbf{v}_0/\mathbf{g}$: base flow and guidance direction separately accessible. *Post-hoc* s: guidance scale adjustable without retraining.

Method	Single Pass	Explicit $\mathbf{v}_0/\mathbf{g}$	Post-hoc s
Two-Pass (NFE = 2N)			
CFG	✗	✓[†]	✓
CFG + Interval	✗	✓[†]	✓
CFG-Zero*	✗	✗[§]	✓
APG	✗	∼[‡]	✓
Autoguidance	✗	✗[¶]	✓
Single-Pass (NFE = N)			
AGD	✓	✗	✓[‖]
GFT	✓	✗	✓[‖]
Model Guidance	✓	✗	✗
ERG	✓	✗	✗
CtrlFlow (Ours)	✓	✓	✓

[†] Via two evaluations, not one pass.
[‡] Projects relative to $\mathbf{v}_{\text{cond}}$, two passes.
[§] Rescales combined update, split not preserved.
[¶] Smaller unconditional net, split not preserved.
[‖] Scale is a network input, one opaque output per pass.

Motivated by this missing intersection, we observe that CFG's two-prediction structure exposes an interpretable interface between a base flow $\mathbf{v}_0$ and a guidance direction $\mathbf{g}$, which should be preserved rather than collapsed into a single opaque prediction. The base flow captures how the model moves toward the data manifold without conditioning, while the guidance direction encodes how the conditioning signal redirects that flow. Existing single-pass efficient methods compress this interface into a single opaque output. To preserve it at one-pass cost, CtrlFlow attaches two lightweight LoRA adapters [Hu et al., 2022], each adding 2.5% parameters, to a frozen backbone and computes both components in a single forward pass. As a result, the guidance scale s multiplies the guidance direction after the forward pass, so s remains adjustable at inference and no retraining is needed. Unlike a scale input, this lets one evaluation yield the guided velocity at every s. In addition to post-hoc scale control, the preserved interface opens a diagnostic channel unavailable to opaque single-pass methods. Norm maps reveal *where* guidance acts spatially, while alignment maps reveal *whether* guidance reinforces or opposes the base flow, and this geometric signal later enables the alignment-based correction in Section 3.4.

Preserving the decomposition is not free, however, because distilling this interface exposes three challenges that require explicit handling.

(i) Training-pressure mismatch. Supervising the two components independently creates a mismatch between training pressure and inference-time error importance within the fixed adapter budget. Inference weights guidance errors by s, but a naive component-wise loss treats both components equally, so the guidance adapter receives weaker gradient signal than its inference-time importance warrants. A unified distillation objective sidesteps this mismatch by supervising only the composed output, which aligns gradient pressure with the composed inference-time error (Section 3.3).

(ii) Error amplification. At high guidance scales with low step counts, small prediction errors in the guidance direction are magnified by s, producing visible artifacts. Scale-adaptive modulation addresses this amplification by targeting three sources of error (Section 3.4). Temporal windowing narrows the guidance-active interval as s grows, while scale dampening reduces the guidance magnitude above a tuned threshold. Alignment-based correction then uses the geometric agreement between base flow and guidance direction, a signal that no opaque single-pass student exposes.

(iii) Dead zones. In our diagnostics, certain (N, s) configurations show large FID degradation that is better explained by temporal under-resolution of the solver than by a globally inaccurate learned velocity field. A scale-dependent time-warp controller redistributes the existing N evaluations near critical trajectory regions without adding forward passes (Section 3.5). With this warp, all observed dead-zone failures except the extreme $(2, 8)$ setting are resolved (Appendix D.13), leaving 24 of 25 configurations within tolerance.

We evaluate on ImageNet 256×256 with the DiT-XL/2 backbone [Peebles and Xie, 2023]. Under the FID tolerance defined in Section 4, these mechanisms recover CFG-match quality while retaining one-pass sampling. CtrlFlow matches CFG quality across all five tested step counts at s=1 and across four of five at s=8. At N=25, wall-clock time falls from 843 ms per image to 437 ms, a $1.93\times$ speedup. Beyond the training range of $s \leq 8$, CtrlFlow remains competitive: it outperforms CFG at s=10 (FID 28.41 versus 31.24) and trails CFG by 3.1 FID at the more extreme s=12 setting in our DiT-XL/2 protocol. Against a scale-conditioned opaque student with the same capacity, teacher, and curriculum (AGD-MultiScale), CtrlFlow is on par at $(25, 2)$ and 0.96 to 4.14 FID lower at the harder tested configurations, matches CFG in 24 of 25 configurations against 20 of 25, and keeps top-1 class accuracy within 2 points of CFG.

Our contributions are organized around preserving and exploiting this interface.

• **Interpretable single-pass guidance interface.** CtrlFlow exposes both $\mathbf{v}_0$ and $\mathbf{g}$ in one forward pass with 5% parameter overhead, while keeping the guidance scale s adjustable at inference without retraining. The dual-adapter architecture and unified distillation objective that enable this interface are described in Sections 3.2 and 3.3.

• **Decomposition-enabled stabilization.** CtrlFlow combines temporal windowing and scale dampening with an alignment-based correction that only the preserved split makes possible, reducing error amplification across the tested range of s. At $(4, 8)$, these modulations lower FID from 73.41 to 63.42 before time warping is applied (Section 3.4, Table 3). These mechanisms also help opaque students, but with every transferable mechanism given to a matched opaque student, CtrlFlow remains 1.3 to 2.1 FID lower (Section 4.3).

• **Dead-zone removal.** Scale-dependent time-grid warping resolves all observed dead-zone failures except $(2, 8)$, leaving 24 of 25 configurations within tolerance without additional forward passes (Section 3.5).

• **CFG-match quality at one-pass cost.** Across the 25-point (N, s) grid on DiT-XL/2, CtrlFlow matches CFG quality in 24 of 25 cases, against 20 of 25 for the matched scale-conditioned opaque student, and the measured N=25 latency shows a $1.93\times$ wall-clock speedup over CFG (Section 4, Table 2). On DiT-L/4, CtrlFlow stays within 0.19 FID of CFG at $(25, 2)$ and outperforms CFG at the tested high-scale settings, and a Stable Diffusion 2.1 pilot adds text conditioning (Section 4.5).

2 Background and Related Works

CFG and its Interface. Because CtrlFlow uses a DiT velocity-prediction backbone [Peebles and Xie, 2023], we write diffusion sampling in the flow-matching form [Lipman et al., 2023], where a learned velocity field transports noise to data along an ODE. Classifier-Free Guidance (CFG) [Ho and Salimans, 2021] steers conditional generation by interpolating unconditional and conditional

predictions,

$$\mathbf{v}_{\text{cfg}} = \mathbf{v}_{\text{uncond}} + s \cdot (\mathbf{v}_{\text{cond}} - \mathbf{v}_{\text{uncond}}), \tag{1}$$

where s controls guidance strength. Equivalently, CFG computes $\mathbf{v}_0 + s\,\mathbf{g}$, where $\mathbf{v}_0 \triangleq \mathbf{v}_{\text{uncond}}$ is the base flow and $\mathbf{g} \triangleq \mathbf{v}_{\text{cond}} - \mathbf{v}_{\text{uncond}}$ is the guidance direction. This decomposition reveals an interpretable interface between unconditional generation and conditioning adjustment, but CFG obtains both components through two evaluations per step, doubling the forward-pass count.

Efficient Guidance. Existing efficiency methods mostly split between lowering the number of evaluations and preserving CFG's separate outputs. Single-pass methods such as Plug-and-Play Distillation [Hsiao et al., 2024], AGD [Jensen and Sadat, 2025], Guidance-Free Training (GFT) [Chen et al., 2025a], Model Guidance [Tang et al., 2025], and ERG [Ifriqi et al., 2025] reduce per-step cost by avoiding a separate unconditional evaluation, often through a compressed guide or training-time guidance surrogate. AGD and GFT take the guidance strength as a network input, so one checkpoint covers many scales, but each pass still returns one opaque guided velocity at one scale. Two-pass refinements such as APG [Sadat et al., 2025a], CFG-Zero* [Fan et al., 2025], Autoguidance [Karras et al., 2024], Dynamic Negative Guidance [Koulischer et al., 2024], and ICG [Sadat et al., 2025b] preserve or improve CFG quality, but they retain the $2N$ evaluation cost. Thus, prior work either preserves the CFG interface at two-pass cost or reduces the pass count by exposing only a single opaque guided output.

Schedules and Training Pressure. This interface gap also affects distillation and sampling design. Distillation methods [Salimans and Ho, 2022, Meng et al., 2023, Song et al., 2023, Luo et al., 2024] collapse the two components into a single composed output, removing the separate guidance direction that CtrlFlow needs for post-hoc control and diagnostics. Guidance-window and solver-schedule methods [Kynkäänniemi et al., 2024, Malarz et al., 2025, Lu et al., 2022, Sabour et al., 2024, Chen et al., 2024] instead modify when or where guidance is applied, but they do not adapt an exposed guidance direction across scales. Once a student is asked to output both $\mathbf{v}_0$ and $\mathbf{g}$, it also faces a training-pressure mismatch: a naive dual loss treats the two adapter errors equally even though inference amplifies guidance errors by s. CtrlFlow responds by preserving the $\mathbf{v}_0/\mathbf{g}$ interface in one pass, aligning distillation with the composed inference-time error, and using scale-aware modulation and time-grid warping to adapt that interface across guidance scales.

Section 3 defines the decomposition formally, implements it with dual adapters, trains it with unified distillation, and introduces the inference-time mechanisms that stabilize it across scales and step counts. Appendix A provides an extended discussion of all related work.

3 Methodology

CtrlFlow distills CFG into a single-pass student that preserves separate base-flow and guidance-direction outputs rather than emitting one opaque guided velocity. Figure 1 summarizes this design. The section first formalizes this output interface and implements it with dual adapters (Sections 3.1 and 3.2). Because the resulting two-output student can still misallocate its limited adapter capacity (the training-pressure mismatch from Section 1), we train it with unified multi-scale distillation (Section 3.3). Finally, two inference-time mechanisms stabilize sampling: scale-adaptive modulation, whose alignment gate reuses the preserved interface, damps high-scale guidance errors, and time-grid warping adapts the solver schedule to the guidance regime (Sections 3.4 and 3.5).

3.1 Velocity Decomposition

As shown in Section 2, CFG can be written as $\mathbf{v}_0 + s\,\mathbf{g}$, where $\mathbf{v}_0$ is the base flow and $\mathbf{g}$ is the guidance direction. We now formalize this split as the output interface that the compressed student must preserve. Given a noisy sample $\mathbf{x}_t$, conditioning signal c, and timestep t, the two teacher-side fields are

$$\mathbf{v}_0(\mathbf{x}_t, t) \triangleq \mathbf{v}_\theta(\mathbf{x}_t, \varnothing, t), \tag{2}$$

$$\mathbf{g}(\mathbf{x}_t, c, t) \triangleq \mathbf{v}_\theta(\mathbf{x}_t, c, t) - \mathbf{v}_\theta(\mathbf{x}_t, \varnothing, t), \tag{3}$$

so the teacher's guided velocity at any scale s is

$$\mathbf{v} = \mathbf{v}_0 + s \cdot \mathbf{g}, \tag{4}$$

with the linear dependence on s explicit. Although this decomposition is algebraically identical to standard CFG, treating the pair $(\mathbf{v}_0, \mathbf{g})$ as the student's required output interface is what enables the

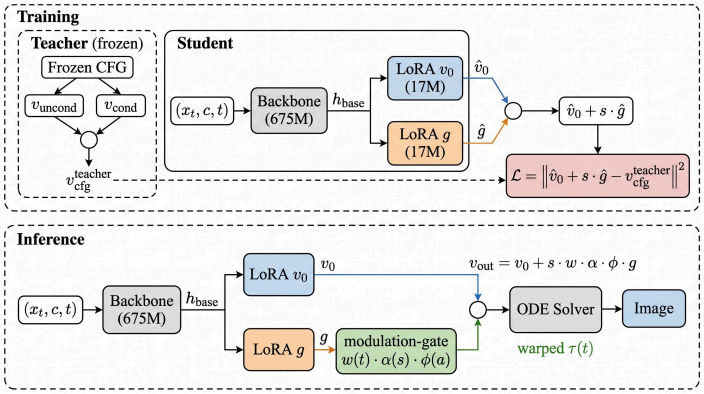

Figure 1: CtrlFlow compresses CFG to one pass while preserving separate $\mathbf{v}_0$ and $\mathbf{g}$ outputs. **Top.** A frozen CFG teacher provides the target $\mathbf{v}_{\text{cfg}}^{\text{teacher}}$. The student matches it via a unified loss on $\hat{\mathbf{v}}_0 + s \cdot \hat{\mathbf{g}}$. **Bottom.** At inference, one backbone pass produces $\mathbf{v}_0$ and $\mathbf{g}$ through two LoRA adapters. The scale s multiplies $\mathbf{g}$ post-hoc. Three modulation gates (w, α, φ) stabilize guidance, and a warped time grid concentrates evaluations near high-curvature regions.

mechanisms that follow: the guidance scale s can be applied after the forward pass, the norm $\|\mathbf{g}\|$ can be inspected spatially, and the alignment between $\mathbf{v}_0$ and $\mathbf{g}$ can drive inference-time gating. In the compressed student, $\hat{\mathbf{v}}_0$ and $\hat{\mathbf{g}}$ (hatted to distinguish student predictions from teacher-defined fields) are learned estimates of these teacher fields produced from a single shared forward pass that receives $(\mathbf{x}_t, c, t)$. The unconditional teacher evaluation that defines $\mathbf{v}_0$ is used only for supervision and is not repeated at inference. Below, hats appear when emphasizing training-time predictions. After training, we write the deployed student outputs as $\mathbf{v}_0$ and $\mathbf{g}$ when no confusion with teacher fields arises.

Interpretable Diagnostics. Because the student preserves both fields as separate outputs, the decomposition exposes two quantities unavailable to single-output methods. The norm $\|\mathbf{g}\|$ measures the magnitude of the conditioning-induced adjustment at each spatial location and timestep: large values indicate strong redirection, while small values mark effectively unconditional regions. The alignment $a \triangleq \cos(\mathbf{v}_0, \mathbf{g})$ quantifies compatibility between the base flow and the conditioning adjustment. When $a \approx 1$, conditioning reinforces the base flow, and when $a < 0$, conditioning opposes it, producing a diagnostic signal for potential artifacts. Section 3.4 uses a as an inference-time modulation signal, and Section 4.4 visualizes both quantities on generated samples. These diagnostics are only useful if compression keeps both fields accessible, which the next subsection implements with two explicit adapter branches.

3.2 Dual-Adapter Architecture

Implementing the decomposition requires an architecture that produces both $\mathbf{v}_0$ and $\mathbf{g}$ in a single forward pass with minimal parameter overhead.

We freeze the pretrained diffusion backbone and attach two Low-Rank Adaptation (LoRA) adapters [Hu et al., 2022], one for each component. Each adapter computes a low-rank residual from the backbone's hidden representations,

$$\mathbf{r} = \mathbf{B}\mathbf{A}\,\mathbf{h}_{\text{base}}, \tag{5}$$

where $\mathbf{A} \in \mathbb{R}^{r \times d}$ and $\mathbf{B} \in \mathbb{R}^{d \times r}$ with rank $r \ll d$. The two branches do not keep separate hidden states. The frozen backbone computes one hidden-state stream per step, both adapters read it at the attention projections, and each branch accumulates its low-rank residuals into its own velocity-shaped output instead of writing them back, so backbone computation runs once per step and NFE $= N$ is exact. The guided velocity is then assembled as $\mathbf{v} = \mathbf{v}_0 + s \cdot \mathbf{g}$ per Equation 4. Because s multiplies $\mathbf{g}$ *after* the forward pass, it remains adjustable at inference without retraining. The full interface is thus preserved with only 2.5% additional parameters per adapter (5% total, rank r=64).

The architecture makes the two fields available, but does not determine how training pressure is distributed across them. A component-wise loss can underweight guidance errors that inference amplifies by s, while composed-output training at a single scale leaves the split underdetermined. Section 3.3 addresses both risks with a unified multi-scale distillation objective.

3.3 Unified Distillation

Training-Pressure Mismatch. The dual-adapter architecture exposes both $\mathbf{v}_0$ and $\mathbf{g}$, but a naive component-wise objective would optimize the wrong error. If the two outputs are supervised independently, base-flow and guidance residuals receive equal loss weight even though inference uses the composed velocity $\hat{\mathbf{v}}_0 + s \cdot \hat{\mathbf{g}}$. A residual error in $\hat{\mathbf{g}}$ is therefore magnified by s at sampling time, whereas a residual error in $\hat{\mathbf{v}}_0$ enters with unit weight. Because the backbone is frozen and each adapter has fixed rank, this mismatch determines which residuals receive training pressure within a fixed budget. At large s, equal component weighting under-prioritizes the guidance errors most likely to produce artifacts. Appendix B.1 formalizes this capacity-allocation analysis.

Unified Distillation Objective. To align training with the deployed interface, CtrlFlow supervises the composed output rather than the two components independently. For each training tuple $(\mathbf{x}_t, c, t, s)$, a frozen CFG teacher provides the two-pass target $\mathbf{v}_{\text{cfg}}^{\text{teacher}}$, and the student minimizes

$$\mathcal{L} = \mathbb{E}\left[\left\|\hat{\mathbf{v}}_0 + s \cdot \hat{\mathbf{g}} - \mathbf{v}_{\text{cfg}}^{\text{teacher}}\right\|^2\right]. \tag{6}$$

This objective gives both adapters the same composed residual while the guidance branch inherits the chain-rule factor s. As a result, high-scale examples place stronger pressure on $\hat{\mathbf{g}}$: Appendix B.1 shows that the guidance adapter's squared gradient norm carries a factor s^2 relative to the base adapter, and verifies empirically the Jacobian-comparability condition behind this factor. The loss therefore allocates capacity according to inference-time error importance without a manually tuned component balance.

Identifiability and Curriculum. Composed-output supervision also creates a uniqueness question: at a single guidance scale, many $(\mathbf{v}_0, \mathbf{g})$ pairs can produce the same sum. Scale variation resolves this degeneracy. Under realizability, Appendix B.1 proves that training on at least two distinct values of s for the same input $(\mathbf{x}_t, c, t)$ uniquely recovers the teacher decomposition (Proposition 1), a condition met by sampling s independently of the input. We use this observation as the organizing principle for a three-phase curriculum: Phase 1 first stabilizes the composed prediction at fixed s, Phase 2 expands to $s \sim \mathcal{U}[1, s_{\max}]$ to identify the split, and Phase 3 samples the (N, s) solver regimes used at inference. This schedule introduces scale and discretization difficulty only after the composed prediction is stable, which avoids the convergence instability seen when Phase 1 is skipped and the single-scale overfitting seen when Phase 2 is skipped (Appendix D.10).

Remaining Gap. Unified distillation aligns training pressure with the composed inference-time error, but it does not remove the amplification caused by high guidance scales. At extreme values of s, residual errors in $\hat{\mathbf{g}}$ are multiplied during inference and can produce visible artifacts. Error amplification of this kind is an inference-time phenomenon, and the next subsection provides an inference-time solution.

3.4 Scale-Adaptive Guidance Modulation

As previewed at the close of Section 3.3, the guided velocity $\mathbf{v}_0 + s \cdot \mathbf{g}$ multiplies prediction errors in $\mathbf{g}$ by the scale factor s at inference time. In the high-scale, low-step regime ($s > 4$, $N \leq 4$), even small guidance errors can produce visible artifacts after amplification by s. We address this inference-time error with three lightweight multipliers that target where errors matter, how strongly they are scaled, and whether guidance agrees geometrically with the base flow. The effective guidance weight becomes $s \cdot w(t) \cdot \alpha(s) \cdot \varphi(a)$, yielding the modulated velocity

$$\mathbf{v}_{\text{out}}(t) = \mathbf{v}_0 + s \cdot w(t) \cdot \alpha(s) \cdot \varphi(a) \cdot \mathbf{g}. \tag{7}$$

Writing $\gamma \triangleq w(t)\,\alpha(s)\,\varphi(a) \in (0, 1]$, modulation shrinks the amplified guidance error from $s\,\boldsymbol{\epsilon}_g$ to $s\gamma\,\boldsymbol{\epsilon}_g$ but also moves the output away from the CFG velocity by $s(1-\gamma)\,\mathbf{g}$, a trade-off that Appendix B.2 states exactly.

Temporal Modulation $w(t)$. Prior work [Kynkäänniemi et al., 2024] observed that guidance is most useful in a mid-timestep interval, whereas endpoint guidance can harm quality or add little benefit. We encode this pattern with a smooth window over t whose width narrows as s grows. Low scales therefore retain broad guidance, while high scales suppress endpoint guidance where amplification is most severe. Appendix C gives the exact Gaussian form and hyperparameters. This scale-adaptive schedule extends fixed-interval guidance [Kynkäänniemi et al., 2024] and beta-distribution schedules [Malarz et al., 2025].

Scale Dampening $\alpha(s)$. Even after temporal windowing narrows where guidance is active, high guidance scales still enlarge residual guidance-direction errors. To counteract this global scale growth, a monotonically decreasing function $\alpha(s)$ reduces the effective guidance magnitude as s increases. The function engages only above a threshold s_{thresh}, so dampening leaves low-s generation unchanged. Appendix C specifies the functional form.

Alignment Gating $\varphi(a)$. The explicit decomposition into $\mathbf{v}_0$ and $\mathbf{g}$ reveals per-step geometric relationships unavailable to single-output methods. The alignment $a = \cos(\mathbf{v}_0, \mathbf{g})$ from Section 3.1 measures whether conditioning reinforces or opposes the base flow. We attenuate guidance only when $a < 0$, where guidance opposes the base flow and is more prone to artifacts, and leave well-aligned steps at full strength. Because CtrlFlow makes a available at every step, this correction requires no additional forward passes. APG [Sadat et al., 2025a] performs a related projection, but it requires two passes and decomposes relative to $\mathbf{v}_{\text{cond}}$ rather than the base flow. Appendix C gives our exact gate. Only $\varphi(a)$ requires the split, since any method with a scale knob can mimic $w(t)$ and $\alpha(s)$ (Section 4.3).

Two-Regime Design. This bias is intentional. At or below s_{thresh}, $\alpha(s)$ and the warp of Section 3.5 are inactive and the window is broad, so CtrlFlow tracks CFG closely (within 0.10 FID at $(25, 2)$). Above it, CFG's own dynamics degrade and CtrlFlow departs from them deliberately: applying windowing and dampening to exact two-pass CFG, which has no prediction error, lowers its FID from 53.84 to 46.02 at $(4, 8)$ (Appendix D.12).

Velocity-field modulation alone, however, does not address every failure case. At extreme (N, s) corners, the solver's uniform time grid undersamples the part of the trajectory where the guided velocity changes most rapidly. Section 3.5 addresses this discretization-side failure by redistributing the fixed evaluations.

3.5 Reducing Dead Zones via Time-Grid Warping

Dead Zones. Certain (N, s) configurations produce sharply degraded generation quality even after velocity-field modulation. The baseline DPM-Solver schedule distributes timesteps uniformly across $[0, 1]$, whereas the guided trajectory changes most rapidly in a narrower mid-timestep region. When the fixed grid spends too many evaluations near smoother endpoints and too few in this high-curvature region, the resulting failure is better explained as temporal under-resolution than as another velocity-field error. We call these configurations *dead zones*.

Scale-Dependent Time Warping. To address dead zones without adding evaluations or retraining, CtrlFlow redistributes the existing N solver calls. A monotone schedule map τ transforms the uniform solver indices $\{i/N\}_{i=0}^{N}$ into non-uniform timesteps $\{t_i = \tau(i/N)\}_{i=0}^{N}$. Its dilation weight $\omega(t)$ is chosen to be small in the mid-timestep region and larger near trajectory endpoints, so the solver takes finer steps where guidance-induced curvature is highest and coarser steps where the trajectory is smoother. The warp magnitude increases only above the same high-scale threshold used by the dampening function in Section 3.4, and at lower guidance scales the schedule remains uniform. Appendices B.3 and C give the exact construction, monotonicity proof, and hyperparameter values.

Properties. Because the warping operates at the ODE solver level, orthogonal to the velocity-field modulations in Section 3.4, it requires no additional forward passes and can be precomputed once per (N, s) configuration with negligible cost. Because τ only moves the solver's timesteps, warping adds no bias to the velocity field (Appendix B.3) and applies to any sampler. Together, modulation and warping stabilize *what* velocity is computed and *where* the solver evaluates it, while preserving the one-pass inference interface.

4 Experiments

4.1 Experimental Setup

Backbone, Dataset, and Protocol. We evaluate CtrlFlow on class-conditional ImageNet 256×256 using DiT-XL/2 [Peebles and Xie, 2023] as the backbone. The frozen CFG teacher runs $N_{\text{teacher}} = 250$ steps with DPM-Solver [Lu et al., 2022]. We generate 50K images per configuration following the ADM evaluation suite [Dhariwal and Nichol, 2021] across $N \in \{2, 4, 8, 16, 25\}$ and $s \in \{1.0, 2.0, 4.0, 6.0, 8.0\}$, yielding 25 configurations. Each configuration is evaluated over three independent 50K-image seed lists, and table entries report mean FID-50K. We also report Precision, Recall [Kynkäänniemi et al., 2019], class adherence (top-1 accuracy of a frozen ResNet-50 [He et al., 2015] on the conditioning class), and CFG-Match Accuracy, defined as the percentage of configurations satisfying $\text{FID}_{CtrlFlow} \leq \text{FID}_{\text{CFG}} + \delta$, where $\delta = \max(2.0,\ 0.10 \times \text{FID}_{\text{CFG}})$. Configurations where CtrlFlow improves over CFG are counted as matches. The absolute floor

of 2.0 handles low-FID regimes where sub-unit differences are not visually meaningful, while the 10% relative term scales the tolerance in harder regimes. The criterion is one-sided by design, since modulation deliberately departs from CFG at high s (Section 3.4).

Baselines. We compare against seven methods spanning two-pass refinements and single-pass compression. (1) **CFG** [Ho and Salimans, 2021] (two-pass oracle), (2) **CFG + Interval** [Kynkäänniemi et al., 2024] (two-pass, mid-timestep guidance), (3) **CFG-Zero*** [Fan et al., 2025] (training-free, two-pass), (4) **APG** [Sadat et al., 2025a] (orthogonal projection, two-pass), (5) **Autoguidance** [Karras et al., 2024] (two-pass, smaller unconditional model), (6) **AGD-MultiScale**, the primary single-pass baseline, an opaque adapter following AGD [Jensen and Sadat, 2025] that receives s as an input, trained with $s \sim \mathcal{U}[1, 8]$ at matched capacity (rank 128, 34M parameters) and with the same teacher, curriculum, and optimizer as CtrlFlow, and (7) **AGD (fixed s=2)**, a smaller (14M) variant trained at one scale. Two-pass baselines run official implementations with default hyperparameters, and we train both AGD variants since no DiT-XL/2 AGD checkpoint is public (Appendix E.1). Appendix D.16 also compares against the released GFT checkpoint.

Table 2: FID-50K (↓) on ImageNet 256×256 with DiT-XL/2. Shaded rows use two passes, unshaded rows one. AGD-MultiScale appears in Table 3 and Appendix D.2, and full statistics and grid in Appendices D.6, D.3, and D.1.

Method	NFE	ms/img (N=25)	s=2 (easy)		s=4	s=8 (hard)	
			N=25	N=4	N=8	N=25	N=4
CFG	$2N$	843	2.97	9.24	11.72	21.47	53.84
CFG-Zero*	$2N$	854	2.81	8.93	11.14	20.67	52.67
CFG + Interval	$2N$	∼843	2.84	8.87	11.21	18.23	46.51
APG	$2N$	852	3.21	10.12	12.84	<u>13.56</u>	<u>39.21</u>
Autoguidance	$2N$	891	<u>2.71</u>	<u>8.61</u>	<u>10.83</u>	18.72	48.35
AGD (fixed s=2)	N	432	**3.04**	10.08	25.81	44.92	82.17
CtrlFlow	N	437	3.07	**9.47**	**10.98**	**19.28**	**48.95**

Training Details. Both LoRA adapters use rank $r = 64$ (5% total overhead), and training follows the three-phase curriculum from Section 3.3. The full run takes 48 GPU-hours on 8×A100. Appendices E.3 and C give optimizer settings, phase lengths, LoRA placement, teacher construction, solver settings, and validation protocol. Modulation and warping hyperparameters are tuned on 5K-image validation samples, while all reported FID values use independent 50K-image generations. These are the five configurations of Table 2, and the other 20 grid configurations were never used for selection.

4.2 Main Results

CtrlFlow uses a single forward pass (NFE $= N$), while CFG and the two-pass baselines require NFE $= 2N$. Table 2 reports FID-50K at five representative (N, s) configurations spanning easy to extreme regimes.

At s=2, where the fixed-scale AGD variant is trained, the tested single-pass methods stay close to CFG, so the easy regime is not the main separator. CtrlFlow falls within 0.10 FID at N=25 and within 0.23 at N=4. The stronger tests are scale generalization and high-scale control. The fixed-scale variant degrades once s leaves its training scale (25.81 at $(8, 4)$, 82.17 at $(4, 8)$), and its Recall of 0.14 at $(25, 8)$, against 0.32 for CFG and 0.35 for CtrlFlow, indicates mode dropping (Appendix D.3). Scale-conditioned training removes most of this failure, yet AGD-MultiScale still trails CtrlFlow at every tested configuration, by up to 4.14 FID at $(25, 8)$, and matches CFG in 20 of 25 grid configurations against 24 of 25 (Appendix D.2). CtrlFlow also improves over CFG at $(8, 4)$ (10.98 versus 11.72) and reaches 19.28 and 48.95 at s=8 with half the forward evaluations. Class adherence survives the damping: CtrlFlow's top-1 accuracy stays within 2 points of CFG's at every tested configuration (75.9% versus 77.8% at $(4, 8)$), while AGD-MultiScale falls to 57.3% (Appendix D.5). Against the released GFT checkpoint at matched NFE, CtrlFlow is 0.27 FID worse at s≈1 and 0.11 to 2.12 FID better for s≈2 to 8, with a frozen backbone instead of full fine-tuning (Appendix D.16).

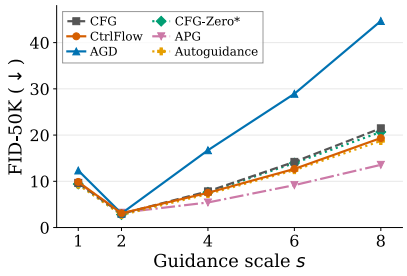

Figure 2: FID-50K vs. scale at N=25. CtrlFlow tracks CFG while the fixed-scale AGD variant ("AGD") drifts past its training scale $s = 2$.

Among two-pass methods, APG gives the best high-s FID through orthogonal projection, but it retains $2N$ cost and does not provide a single-pass base-flow/guidance interface. Among CFG-style methods,

CtrlFlow reaches within 1.05 FID of CFG + Interval at $(25, 8)$ with half the forward evaluations while retaining the decomposition interface.

Figure 2 extends this comparison across the full guidance-scale range at N=25. The fixed-scale AGD curve climbs sharply beyond its training scale of $s = 2$, whereas CtrlFlow tracks CFG with half the forward evaluations. Appendix Figure 10 plots FID against NFE at s=4 and shows the same compute-quality pattern.

Across the full 25-configuration grid (Appendix D.1), CtrlFlow achieves 96% CFG-match accuracy (24 of 25 within δ) and a $1.93\times$ wall-clock speedup at N=25. This match count is stable across three generation seeds (Appendix D.6), and all 20 configurations never used for tuning match except $(2, 8)$. Beyond the training range of $s \leq 8$, CtrlFlow remains competitive, outperforming CFG at s=10 and trailing by 3.1 FID at s=12 (Appendix D.14), and qualitative samples appear in Appendix D.21.

4.3 Ablation Studies

Contribution Isolation. Table 3 separates the value of the explicit decomposition from the benefits of multi-scale training by building up from the core architecture. CtrlFlow-Core uses only the dual-adapter decomposition with unified distillation, omitting modulation and warping. As a controlled comparison, AGD-MultiScale trains a single opaque adapter that receives s as an input, with the same multi-scale curriculum and matched parameter budget (rank 128, 34M parameters). At moderate scales the two are close: AGD-MultiScale reaches FID 11.94 versus CtrlFlow-Core's 11.48 at $(8, 4)$. At the extreme $(4, 8)$, however, AGD-MultiScale achieves 52.87 while CtrlFlow-Core reaches only 73.41, so the decomposition constraint alone does not provide a raw optimization advantage. The picture reverses once the interface is exploited, with CtrlFlow-Full reaching 48.95.

Table 3: Contribution isolation, FID-50K (↓). Core uses dual adapters and unified distillation, and +Mod/+Warp add stabilization. AGD-MultiScale is a matched-budget, scale-conditioned opaque distillation baseline.

Variant	$(25, 2)$	$(8, 4)$	$(4, 8)$
AGD (fixed s=2)	3.04	25.81	82.17
AGD-MultiScale (r=128)	3.09	11.94	52.87
CtrlFlow-Core	3.14	11.48	73.41
CtrlFlow+Mod	3.12	11.21	63.42
CtrlFlow+Warp	3.15	11.35	59.83
CtrlFlow-Full	**3.07**	**10.98**	**48.95**

Interface versus Transferable Mechanisms. Because windowing, dampening, and warping are not specific to the decomposition, we also give them to the baselines, feeding the opaque student a time-varying scale input $s_{\text{eff}}(t) = s\, w(t)\, \alpha(s)$ (Appendix D.12). They improve AGD-MultiScale by 0.7 to 2.3 FID, yet CtrlFlow-Full stays 1.3 to 2.1 FID lower at all four tested configurations, including two never used for tuning, and removing only the alignment gate, the one mechanism that needs $\mathbf{v}_0$ and $\mathbf{g}$, costs 0.6 to 2.4 FID. The decomposition's value is therefore the interface it opens, not a raw quality gap.

Further Ablations. Four additional studies (Appendices D.8 to D.11) confirm that the mechanisms contribute independently and that the three-phase curriculum is important for stable high-scale performance. Unified distillation produces the largest single effect (FID degrades by 19.5 at $(4, 8)$ when removed). The three modulation sub-components are complementary, with each pair outperforming both of its components, and rank 64 sits at the knee of the cost-quality curve. A conditional-field anchor, $\mathbf{v}_{\text{cond}} + (s - 1)\,\mathbf{g}$, matches Core within noise but trails Full by 1.1 to 2.6 FID (Appendix D.20). Dead-zone analysis and modulation/warping visualizations appear in Appendices D.13, D.24, and D.25.

4.4 Decomposition Fidelity and Interpretability

The explicit decomposition into $\mathbf{v}_0$ and $\mathbf{g}$ enables direct inspection of what guidance changes, a capability unavailable to opaque single-pass methods that predict only the composed guided velocity. Before using this channel for visualization, we verify that the learned components are faithful to the teacher decomposition in two ways. Cosine similarity to the teacher-generated ground truth exceeds 0.83 across all timesteps and peaks at 0.97 for $\hat{\mathbf{v}}_0$ at t=0.5 (Appendix D.18). We then test functional equivalence: each learned component replaces its teacher counterpart in a standard two-pass CFG pipeline, and we measure whether generation quality is preserved. Table 4 reports the results.

Table 4: Component swap, FID-50K (↓).

Variant	$(25, 2)$	$(25, 8)$
CFG teacher	2.97	21.47
Swap $\hat{\mathbf{g}}$	3.07	22.31
Swap $\hat{\mathbf{v}}_0$	3.01	21.74
CtrlFlow	3.07	19.28

At $(25, 2)$, swapping in $\hat{\mathbf{g}}$ raises FID by only 0.10, while swapping in $\hat{\mathbf{v}}_0$ raises it by 0.04. At $(25, 8)$, the gaps grow to 0.84 for $\hat{\mathbf{g}}$ and 0.27 for $\hat{\mathbf{v}}_0$ because guidance errors are amplified by s. The full CtrlFlow model reaches 19.28 at $(25, 8)$, outperforming both swap variants and CFG itself. This gain reflects the deliberate high-scale departure from CFG (Appendix B.2.5), not a closer CFG approximation. Having verified that the split is both geometrically and functionally meaningful, we next visualize what the two components reveal.

Figure 3 uses this verified decomposition to visualize where and when guidance acts on a representative ImageNet class (*golden retriever*). The top row shows that guidance is spatially diffuse at high noise but concentrates by later timesteps on class-discriminative regions such as object boundaries, fur texture, and facial features, while background regions receive little signal. The bottom row shows the complementary geometric pattern: alignment between $\mathbf{v}_0$ and $\mathbf{g}$ starts near zero and increases toward 1 as the trajectory converges. This progression is not imposed explicitly. It emerges from unified distillation and supplies the signal used by the alignment gate $\varphi(a)$ in Section 3.4. The same spatial and temporal patterns appear for butterfly and volcano classes in Appendix D.22, suggesting that the learned split is not specific to the representative class shown here. Quantitatively, guidance-norm maps on noised ImageNet validation images point to the annotated object with accuracy 0.81, above Grad-CAM [Selvaraju et al., 2017] (0.74) and on par with the two-pass teacher's maps (0.80) (Appendix D.23).

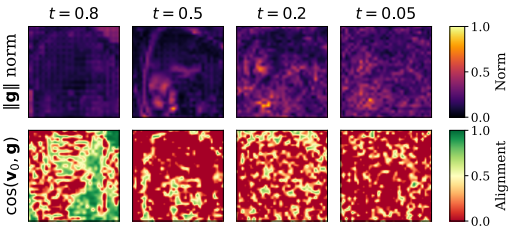

Figure 3: Decomposition maps for *golden retriever*. Guidance norm becomes spatially focused as structure emerges, while $\cos(\mathbf{v}_0, \mathbf{g})$ shifts from near-orthogonal to aligned. Extended classes appear in Appendix D.22.

4.5 Cost and Generalization

The interpretability and stabilization benefits of the decomposition come at modest cost. CtrlFlow adds 34M parameters (5% of DiT-XL/2) and requires 48 GPU-hours of training, but at $N{=}25$ it reduces latency from 843 ms per image for CFG to 437 ms, and Appendix Table 29 compares all methods. The speedup is $1.93\times$ rather than $2\times$ because the adapters and modulation add 15.5 ms to the 421.5 ms backbone pass (Appendix E.2). To test transfer beyond DiT-XL/2, we also train CtrlFlow on DiT-L/4 with the same protocol. On DiT-L/4, CtrlFlow stays within 0.19 FID of CFG at $(25, 2)$ and outperforms CFG at the tested high-scale settings, with full cross-backbone and decomposition-fidelity tables in Appendix D.15. A Stable Diffusion 2.1 [Rombach et al., 2022] text-to-image pilot keeps the ImageNet ordering: at $N{=}25$ and $s{=}7.5$, CtrlFlow reaches COCO FID-10K 10.41 and CLIPScore 0.304, between two-pass CFG (9.82, 0.309) and a scale-conditioned opaque student (11.37, 0.300) (Appendix D.17).

5 Conclusion

CtrlFlow distills CFG into a single forward pass that preserves the base-flow and guidance-direction interface with two LoRA adapters, keeping s adjustable without retraining. With unified distillation, scale-adaptive modulation, and time-grid warping, it matches CFG in 24 of 25 (N, s) configurations with a $1.93\times$ speedup at $N{=}25$ on DiT-XL/2, ahead of a matched scale-conditioned opaque student. Because the same decomposition supports post-hoc scale control, per-step diagnostics, and error-aware stabilization, the results suggest that guidance efficiency and interpretability need not compete.

Limitations. Our main evaluation covers two DiT backbones on class-conditional ImageNet $256{\times}256$, text conditioning is covered only by a single-setting Stable Diffusion 2.1 pilot, and video and higher resolutions remain untested. Modulation hyperparameters were selected on the headline configurations, although the 20 untuned ones show the same pattern (Appendix C), and above s_{thresh} they make CtrlFlow match CFG quality rather than reproduce CFG samples. Decomposition fidelity rests on cosine similarity, component swaps, and localization scores, while Proposition 1 assumes realizability and global zero loss, so finite-sample recovery remains empirical.

Broader Impacts. The interface provides per-step inspection that could support content-safety pipelines, but making guided generation cheaper inherits the risks of the underlying technology, including deepfake misuse. Guidance-aware safety mechanisms monitoring decomposition signals deserve dedicated investigation.

Acknowledgments and Disclosure of Funding

This work is supported in part by the U.S. National Science Foundation under Grants CNS-2247892, CNS-2239672, OAC-2319962, CNS-2326597, and CCF-2449989. We thank Modal for supporting this work through Modal academic research credits, which provided part of the compute used in our experiments. We thank the anonymous reviewers and the area chair for constructive feedback that improved the baselines, analyses, and positioning of this paper. The authors declare no competing interests.

References

Huayu Chen, Kai Jiang, Kaiwen Zheng, Jianfei Chen, Hang Su, and Jun Zhu. Visual generation without guidance. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=gM6aboVgTO.

Huayu Chen, Hang Su, Peize Sun, and Jun Zhu. Toward guidance-free AR visual generation via condition contrastive alignment. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=kGvXIlIVLM.

Yuzhu Chen, Fengxiang He, Shi Fu, Xinmei Tian, and Dacheng Tao. Adaptive time-stepping schedules for diffusion models. In Negar Kiyavash and Joris M. Mooij, editors, *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*, volume 244 of *Proceedings of Machine Learning Research*, pages 685–697. PMLR, 15–19 Jul 2024. URL https://proceedings.mlr.press/v244/chen24c.html.

Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. CFG++: Manifold-constrained classifier free guidance for diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=E77uvbOTtp.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=AAWuCvzaVt.

Weichen Fan, Amber Yijia Zheng, Raymond A. Yeh, and Ziwei Liu. Cfg-zero*: Improved classifier-free guidance for flow matching models, 2025. URL https://arxiv.org/abs/2503.18886.

Pareesa Ameneh Golnari. Lora-enhanced distillation on guided diffusion models, 2023. URL https://arxiv.org/abs/2312.06899.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A reference-free evaluation metric for image captioning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7514–7528, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.595. URL https://aclanthology.org/2021.emnlp-main.595/.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Yi-Ting Hsiao, Siavash Khodadadeh, Kevin Duarte, Wei-An Lin, Hui Qu, Mingi Kwon, and Ratheesh Kalarot. Plug-and-play diffusion distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13743–13752, 2024.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Tariq Berrada Ifriqi, Adriana Romero-Soriano, Michal Drozdzal, Jakob Verbeek, and Karteek Alahari. Entropy rectifying guidance for diffusion and flow models, 2025. URL https://arxiv.org/abs/2504.13987.

Cristian Perez Jensen and Seyedmorteza Sadat. Efficient distillation of classifier-free guidance using adapters. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=uMz8FiiW01.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=bg6fVPVs3s.

Felix Koulischer, Johannes Deleu, Gabriel Raya, Thomas Demeester, and Luca Ambrogioni. Dynamic negative guidance of diffusion models: Towards immediate content removal. In *Neurips Safe Generative AI Workshop 2024*, 2024. URL https://openreview.net/forum?id=A2UpLzwDz5.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=nAIhvNy15T.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

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

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: a fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2024. URL https://openreview.net/forum?id=duBCwjb68o.

Dawid Malarz, Artur Kasymov, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. Classifier-free guidance with adaptive scaling, 2025. URL https://arxiv.org/abs/2502.10574.

Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14297–14306, 2023. doi: 10.1109/CVPR52729.2023.01374.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers . In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182, Los Alamitos, CA, USA, October 2023. IEEE Computer Society. doi: 10.1109/ICCV51070.2023.00387. URL https://doi.ieeecomputersociety.org/10.1109/ICCV51070.2023.00387.

Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models, 2018. URL https://arxiv.org/abs/1806.07421.

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)*, pages 10674–10685, Los Alamitos, CA, USA, June 2022. IEEE Computer Society. doi: 10.1109/CVPR52688.2022.01042. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52688.2022.01042.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your steps: optimizing sampling schedules in diffusion models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Seyedmorteza Sadat, Otmar Hilliges, and Romann M. Weber. Eliminating oversaturation and artifacts of high guidance scales in diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=e2ONKX6qzJ.

Seyedmorteza Sadat, Manuel Kansy, Otmar Hilliges, and Romann M. Weber. No training, no problem: Rethinking classifier-free guidance for diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=b3CzCCCILJ.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. doi: 10.1109/ICCV.2017.74.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Zhicong Tang, Jianmin Bao, Dong Chen, and Baining Guo. Diffusion models without classifier-free guidance, 2025. URL https://arxiv.org/abs/2502.12154.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *Int. J. Comput. Vision*, 126(10):1084–1102, October 2018. ISSN 0920-5691. doi: 10.1007/s11263-017-1059-x. URL https://doi.org/10.1007/s11263-017-1059-x.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process, 2026. URL https://arxiv.org/abs/2509.16117.

Appendices

A Extended Related Work **16**

B Theoretical Analysis **17**

B.1 Capacity Allocation Analysis 17

B.1.1 Setup and Notation 17

B.1.2 Objective Mismatch under the Naive Loss 17

B.1.3 Unified Loss Resolves the Mismatch 18

B.1.4 Scale-Dependent Capacity Allocation 18

B.1.5 Decomposition Identifiability under Scale Variation 18

B.2 Error Amplification Analysis 19

B.2.1 Error Model 19

B.2.2 Per-Step Error Bound 19

B.2.3 Trajectory Error Accumulation 20

B.2.4 How Each Modulation Mechanism Reduces Amplification 20

B.2.5 Bias-Error Trade-off and Two-Regime Design 21

B.3 Time-Warp Properties 21

B.3.1 Monotonicity and Endpoint Preservation 21

B.3.2 Discretization Error Reduction 22

B.3.3 Computational Cost 22

C Hyperparameter Specification **23**

C.1 Modulation Functions (Section 3.4) 23

C.2 Time-Warp Parameters (Section 3.5) 23

C.3 Shared Parameters 24

C.4 Summary Table 24

C.5 Sensitivity Discussion 24

D Extended Experimental Results **25**

D.1 Full Configuration Grid 25

D.2 Scale-Conditioned Opaque Baseline 25

D.3 Precision and Recall 26

D.4 Inception Score 26

D.5 Class Adherence 26

D.6 FID Standard Deviations 27

D.7 Training Convergence 27

D.8 Component Ablation 27

D.9 Per-Modulation Sub-Ablation 28

D.10 Curriculum Ablation 28
D.11 LoRA Rank Sensitivity 28
D.12 Mechanism Transfer to Baselines 29
D.13 Dead-Zone Analysis 29
D.14 Scale Extrapolation 30
D.15 Cross-Backbone Generalization 30
D.16 Comparison with Guidance-Free Training 31
D.17 Text-to-Image Pilot 31
D.18 Decomposition Fidelity 32
D.19 Decomposition Functional Equivalence 32
D.20 Anchor Ablation: Unconditional versus Conditional Base 32
D.21 Qualitative Comparison and Visualizations 34
D.22 Extended Interpretability Maps 35
D.23 Quantitative Localization of Guidance Maps 36
D.24 Modulation and Warping Visualization 37
D.25 Time-Warp Visualization 37
D.26 FID vs. NFE 38

E Reproducibility **38**

E.1 Baseline Protocol 38
E.2 Computational Cost 39
E.3 Implementation Details 39

A Extended Related Work

This appendix provides additional detail on the related work summarized in Section 2.

Classifier-Free Guidance. Dhariwal and Nichol [2021] introduced classifier guidance, which steers sampling toward higher-fidelity outputs by injecting gradients from a separately trained classifier at the cost of reduced sample diversity. CFG [Ho and Salimans, 2021] removed the need for an external classifier by interpolating between conditional and unconditional score estimates. CFG++ [Chung et al., 2025] recasts guidance as manifold-constrained optimization and shows that standard CFG displaces samples from the learned data manifold when the guidance scale grows large.

Single-Pass Methods. Plug-and-Play Diffusion Distillation [Hsiao et al., 2024] trains a lightweight external guide at 1% of the base model's parameter count, keeping the base frozen and cutting CFG cost by roughly half. Building on this idea, AGD [Jensen and Sadat, 2025] distills CFG into low-rank adapters on a frozen backbone and gives the adapter the guidance scale as an extra input, sampled over a range during training, so one checkpoint supports post-hoc scale changes. Its output is still a single opaque guided velocity. Guidance-Free Training [Chen et al., 2025a] reaches a similar one-model interface by fine-tuning the full backbone with a temperature input that sets the guidance strength. Because the strength enters as an input in both methods, each forward pass returns one guided prediction at one strength, and recovering the guidance direction or a second strength costs further forward passes. GFT's temperature also covers only non-negative scales. CtrlFlow instead applies the scale after the forward pass, and Appendix D.16 compares the two directly. GFT generalizes Condition Contrastive Alignment [Chen et al., 2025b], which fine-tunes autoregressive visual models to sample without guidance, and guidance-free training has since been adopted downstream, for example in the CFG-free reinforcement-learning pipeline of DiffusionNFT [Zheng et al., 2026]. The guidance-free methods we are aware of keep GFT's input-side interface and return one opaque output per pass, so none of them exposes the base flow and guidance direction separately. Model Guidance [Tang et al., 2025] sidesteps CFG altogether by folding the posterior probability of conditions directly into the training loss. Entropy Rectifying Guidance [Ifriqi et al., 2025] substitutes attention-based energy signals for the unconditional prediction. Independent Condition Guidance [Sadat et al., 2025b] conditions on an independent signal instead of an unconditional one, which removes the need for unconditional training yet still requires two forward passes.

Two-Pass Improvements. APG [Sadat et al., 2025a] decomposes the guidance update into parallel and orthogonal components relative to the conditional prediction. It selectively down-weights the parallel component to suppress oversaturation artifacts. CFG-Zero* [Fan et al., 2025] refines initialization and rescaling specifically for flow matching models. Autoguidance [Karras et al., 2024] swaps the unconditional model for a separately trained, lower-capacity network, incurring additional storage and training cost. Dynamic Negative Guidance [Koulischer et al., 2024] ties the guidance scale to the posterior probability at each state. Guidance thereby fades in regions where it offers no measurable benefit.

Distillation and Step Reduction. Progressive distillation [Salimans and Ho, 2022] repeatedly halves the number of sampling steps. Meng et al. [2023] extended this approach to CFG-guided models, reaching 1 to 4 step generation. Their scale-conditioned student can adjust s freely at inference time. Consistency models [Song et al., 2023] learn direct mappings from noise to data in a single pass. Latent consistency models [Luo et al., 2024] bring this paradigm into the latent diffusion setting. Golnari [2023] explored combining LoRA with guidance distillation. Each of these methods treats the guided velocity field as an opaque regression target.

Guidance and Sampling Schedules. Kynkäänniemi et al. [2024] showed that guidance degrades quality at high noise levels and adds nothing near convergence. Only a middle noise interval benefits from guidance, and restricting it to that window improved ImageNet-512 FID from 1.81 to 1.40. Malarz et al. [2025] generalized this finding with beta-distribution-shaped schedule curves. Both approaches rely on fixed schedules that do not vary with s. DPM-Solver [Lu et al., 2022] leverages the semi-linear ODE structure of diffusion models for efficient high-order numerical solving. Sabour et al. [2024] demonstrated that optimizing the timestep schedule itself yields substantial gains in few-step regimes. Chen et al. [2024] proposed gradient-based optimization of discretization points for similar improvements. All of these methods assume a fixed guidance configuration and do not adapt

their behavior to the guidance scale s. Taken together, prior work either preserves CFG's interface at two-pass cost or reduces the pass count by treating the guided velocity as opaque. CtrlFlow targets the remaining intersection by preserving the interface in a single pass and adapting its use across scale and time.

B Theoretical Analysis

This appendix collects the formal analyses that justify the optimization and control structure of CtrlFlow. Section B.1 analyzes capacity allocation under the unified loss. Section B.2 formalizes the error amplification mechanism and derives bounds for the modulation mechanisms. Section B.3 proves the formal properties of the time-grid warping.

Scope. The analyses in Sections B.1 through B.3 do not prove finite-sample convergence, global optimization of LoRA adapters, or optimality of the specific modulation hyperparameters. The identifiability result (Proposition 1) assumes realizability and global zero loss. The capacity-allocation ratio relies on a Jacobian-comparability condition, which we measure rather than assume (Section B.1), and the modulation analysis separates the estimation error that modulation suppresses from the bias that it deliberately introduces (Section B.2.5). Empirical ablations in Section 4 validate the actual training and inference setting.

B.1 Capacity Allocation Analysis

This section analyzes the capacity misallocation under the naive dual-loss (Section 3.3) and shows that the unified distillation objective resolves it. All gradients are with respect to the trainable adapter parameters only, as the backbone is frozen throughout.

B.1.1 Setup and Notation

The pretrained backbone is frozen and produces shared features $\mathbf{h}_{\text{base}}(\mathbf{x}_t, c, t)$. Let $\theta = (\mathbf{A}_0, \mathbf{B}_0, \mathbf{A}_g, \mathbf{B}_g)$ denote the trainable LoRA adapter parameters. The base-flow adapter computes $\hat{\mathbf{v}}_0 = f_0(\mathbf{h}_{\text{base}}; \mathbf{A}_0, \mathbf{B}_0)$, while the guidance adapter computes $\hat{\mathbf{g}} = f_g(\mathbf{h}_{\text{base}}; \mathbf{A}_g, \mathbf{B}_g)$. The teacher provides targets $\mathbf{v}_0^*$ and $\mathbf{g}^*$ from two forward passes of the frozen CFG model. We write the per-sample errors as $\boldsymbol{\delta}_0 \triangleq \hat{\mathbf{v}}_0 - \mathbf{v}_0^*$ and $\boldsymbol{\delta}_g \triangleq \hat{\mathbf{g}} - \mathbf{g}^*$.

Because the backbone is frozen, $\mathbf{h}_{\text{base}}$ is a fixed input to both adapters for any given sample. The two adapter parameter sets $(\mathbf{A}_0, \mathbf{B}_0)$ and $(\mathbf{A}_g, \mathbf{B}_g)$ are disjoint. Gradients with respect to one adapter's parameters do not directly affect the other. The interaction arises indirectly, since both adapters operate on the same frozen features and must represent their respective targets within the same feature space under a shared rank constraint.

B.1.2 Objective Mismatch under the Naive Loss

Claim 1 (Capacity misallocation under $\mathcal{L}_{\text{naive}}$)**.** *Consider the naive dual-loss $\mathcal{L}_{naive} = \|\boldsymbol{\delta}_0\|^2 + \|\boldsymbol{\delta}_g\|^2$. Because the backbone is frozen, the gradient decomposes into two independent adapter updates,*

$$\nabla_{\theta_0} \mathcal{L}_{naive} = 2\, \boldsymbol{\delta}_0^\top \nabla_{\theta_0} \hat{\mathbf{v}}_0, \quad \nabla_{\theta_g} \mathcal{L}_{naive} = 2\, \boldsymbol{\delta}_g^\top \nabla_{\theta_g} \hat{\mathbf{g}}, \tag{8}$$

where $\theta_0 = (\mathbf{A}_0, \mathbf{B}_0)$ and $\theta_g = (\mathbf{A}_g, \mathbf{B}_g)$. Each adapter receives equal implicit priority. The loss weights both $\|\boldsymbol{\delta}_0\|^2$ and $\|\boldsymbol{\delta}_g\|^2$ by 1. At inference, however, the composed output is $\hat{\mathbf{v}}_0 + s\,\hat{\mathbf{g}}$. An error $\boldsymbol{\delta}_g$ in the guidance component contributes $s\,\boldsymbol{\delta}_g$ to the composed error, while $\boldsymbol{\delta}_0$ contributes only $\boldsymbol{\delta}_0$. The naive loss ignores this asymmetry. Because the adapters have fixed ranks, the loss cannot move parameters between branches. Instead, it determines which residual errors receive stronger gradient pressure within the fixed dual-adapter budget. Equal weighting provides no mechanism to direct that pressure toward the component whose errors matter s times more at inference.

Discussion. The preceding claim is a training-pressure argument, not a gradient-conflict argument in the classical multi-task sense [Yu et al., 2020]. Because the backbone is frozen and the two adapter parameter sets are disjoint, gradients do not directly interfere. The mismatch arises because the naive loss weights both adapter errors equally, whereas inference amplifies guidance errors by s. Within the fixed dual-adapter budget, the naive loss provides no signal to direct gradient pressure toward the component that matters more at inference.

B.1.3 Unified Loss Resolves the Mismatch

Claim 2 (Unified gradient as a coherent update)**.** *Under the unified loss* $\mathcal{L} = \|\hat{\mathbf{v}}_0 + s\,\hat{\mathbf{g}} - \mathbf{v}_{cfg}^{teacher}\|^2$*, the gradients with respect to the adapter parameters are*

$$\nabla_{\theta_0}\mathcal{L} = 2\,\boldsymbol{\delta}_{comp}^{\top}\nabla_{\theta_0}\hat{\mathbf{v}}_0, \quad \nabla_{\theta_g}\mathcal{L} = 2s\,\boldsymbol{\delta}_{comp}^{\top}\nabla_{\theta_g}\hat{\mathbf{g}}, \tag{9}$$

where $\boldsymbol{\delta}_{comp} = \hat{\mathbf{v}}_0 + s\,\hat{\mathbf{g}} - \mathbf{v}_{cfg}^{teacher}$ *is the composed error. Both adapters are driven by the same error signal* $\boldsymbol{\delta}_{comp}$*. The guidance adapter's gradient carries an additional factor of* s*, so its updates are scaled in proportion to the downstream impact of guidance errors. There is no decomposition into independently weighted objectives and hence no mismatch between training priority and inference importance.*

Proof sketch. Differentiating $\mathcal{L}$ with respect to θ_g produces the chain-rule factor $\nabla_{\theta_g}(s\,\hat{\mathbf{g}}) = s\,\nabla_{\theta_g}\hat{\mathbf{g}}$. The factor s appears because the loss directly penalizes the composed output in which $\hat{\mathbf{g}}$ is scaled by s. Differentiating with respect to θ_0 yields no such factor, since $\hat{\mathbf{v}}_0$ enters the composed output with unit weight.

B.1.4 Scale-Dependent Capacity Allocation

Claim 3 (s^2 squared-gradient weighting)**.** *Let* $J_0 \triangleq \nabla_{\theta_0}\hat{\mathbf{v}}_0$ *and* $J_g \triangleq \nabla_{\theta_g}\hat{\mathbf{g}}$ *denote the adapter Jacobians. Under the unified loss, the squared gradient norms for each adapter satisfy*

$$\|\nabla_{\theta_0}\mathcal{L}\|^2 = 4\,\|\boldsymbol{\delta}_{comp}^{\top}J_0\|^2, \quad \|\nabla_{\theta_g}\mathcal{L}\|^2 = 4s^2\,\|\boldsymbol{\delta}_{comp}^{\top}J_g\|^2, \tag{10}$$

so their ratio is exactly

$$\frac{\|\nabla_{\theta_g}\mathcal{L}\|^2}{\|\nabla_{\theta_0}\mathcal{L}\|^2} = s^2 \cdot \frac{\|\boldsymbol{\delta}_{comp}^{\top}J_g\|^2}{\|\boldsymbol{\delta}_{comp}^{\top}J_0\|^2}. \tag{11}$$

When the two Jacobians respond comparably to the shared composed error, $\|\boldsymbol{\delta}_{comp}^{\top}J_g\| \approx \|\boldsymbol{\delta}_{comp}^{\top}J_0\|$*, the squared-gradient ratio is* $\approx s^2$*, which corresponds to a gradient-magnitude ratio of* $\approx s$*. At* $s = 1$*, both adapters receive comparable gradient signal. At* $s = 8$*, the guidance adapter's squared gradient norm is ideally* $s^2 = 64$ *times that of the base adapter (*52.7 *as measured below), a gradient magnitude roughly 7 to 8 times larger, which increases training pressure on the component whose errors are amplified most at inference.*

Proof sketch and empirical check. The s^2 factor follows from squaring the chain-rule factor s in $\nabla_{\theta_g}\mathcal{L}$, so Equation 11 holds without assumptions. The comparability condition is not implied by equal rank alone, because the two adapters map the same composed error through different learned weights, so we measure it rather than assume it. On matched training states, the measured squared-gradient ratio is 4.3, 15.1, and 52.7 at $s = 2$, 4, and 8, against $s^2 = 4$, 16, and 64. The Jacobian ratio in Equation 11 therefore stays between 0.82 and 1.08 across the training range, and the s^2 approximation holds to within 18%.

This s^2 weighting is the formal mechanism behind the claim in Section 3.3 that the chain-rule factor s scales the guidance-adapter gradient. Unlike gradient surgery approaches [Yu et al., 2020] that project conflicting task gradients post-hoc, the unified loss produces a coherent training signal. It allocates training pressure proportional to the downstream impact of each component's error.

Scope of this analysis. The s^2 weighting shows that the unified loss aligns training pressure with inference-time error importance. Whether the learned decomposition uniquely recovers the teacher's $\mathbf{v}_0$ and $\mathbf{g}$ is a separate question. Proposition 1 below addresses it under a realizability assumption, and Section 4.4 validates it empirically.

B.1.5 Decomposition Identifiability under Scale Variation

The preceding analysis shows that the unified loss allocates training pressure proportionally to inference-time importance. A separate question is whether the loss has a unique minimizer that recovers the true decomposition. The following proposition shows that it does, provided the training distribution includes variation in s.

Proposition 1 (Identifiability under scale variation)**.** *Let* $\mathcal{F}_0$ *and* $\mathcal{F}_g$ *be function classes for the base and guidance adapters. Suppose:*

(i) For almost every input $(\mathbf{x}_t, c, t)$, the training distribution of s given that input has support containing at least two distinct values $s_1 \neq s_2$.

(ii) There exist $f_0^* \in \mathcal{F}_0$ and $f_g^* \in \mathcal{F}_g$ such that $\mathbf{v}_{\text{cfg}}(\mathbf{x}_t, c, t, s) = f_0^*(\mathbf{h}) + s \cdot f_g^*(\mathbf{h})$ for all $(\mathbf{x}_t, c, t, s)$.

Then the global minimizer of $\mathcal{L} = \mathbb{E}_s\left[\|f_0(\mathbf{h}) + s \cdot f_g(\mathbf{h}) - \mathbf{v}_{\text{cfg}}\|^2\right]$ is unique: $f_0 = f_0^*$ and $f_g = f_g^*$.

Proof. At the global minimum, $\mathcal{L} = 0$. For any sample $(\mathbf{x}_t, c, t)$, define $\boldsymbol{\delta}_0 = f_0(\mathbf{h}) - f_0^*(\mathbf{h})$ and $\boldsymbol{\delta}_g = f_g(\mathbf{h}) - f_g^*(\mathbf{h})$. The residual $r(s) = \boldsymbol{\delta}_0 + s \cdot \boldsymbol{\delta}_g$ must equal zero for all s in the support. Evaluating at s_1 and s_2:

$$\boldsymbol{\delta}_0 + s_1 \cdot \boldsymbol{\delta}_g = \mathbf{0}, \tag{12}$$

$$\boldsymbol{\delta}_0 + s_2 \cdot \boldsymbol{\delta}_g = \mathbf{0}. \tag{13}$$

Subtracting yields $(s_1 - s_2) \cdot \boldsymbol{\delta}_g = \mathbf{0}$. Since $s_1 \neq s_2$, we have $\boldsymbol{\delta}_g = \mathbf{0}$, and substituting back gives $\boldsymbol{\delta}_0 = \mathbf{0}$. □

Interpretation. Condition (i) must hold for each input, not only over the whole training set, since a schedule that paired each input with a single scale would leave the split undetermined input by input. Phases 2 and 3 of the curriculum satisfy it because they sample s from a continuous distribution independently of $(\mathbf{x}_t, c, t)$. Condition (ii) is the realizability assumption, requiring that the adapter function classes be expressive enough to represent the true components. In practice, low-rank adapters provide an approximation, and the decomposition is correspondingly approximate. The proposition shows that the unified loss has the correct inductive bias: when the true components are representable and the training distribution includes scale variation, the global minimizer recovers the teacher decomposition. In finite-capacity LoRA adapters, this result should be read as an inductive-bias argument rather than a convergence guarantee. The decomposition is correspondingly approximate, and Section 4.4 validates it empirically.

This result also explains why Phase 1 (fixed $s = s_{\text{target}}$) cannot identify the decomposition. At a single scale, the loss constrains only the sum $f_0(\mathbf{h}) + s_{\text{target}} \cdot f_g(\mathbf{h})$, leaving infinitely many valid (f_0, f_g) pairs. Scale variation in Phases 2 and 3 breaks this degeneracy.

B.2 Error Amplification Analysis

This section formalizes the error amplification mechanism described in Section 3.4 and derives bounds that justify the three modulation mechanisms. It then states the exact decomposition of the modulated output against the CFG velocity, which separates the estimation error that modulation suppresses from the bias that it deliberately introduces (Section B.2.5).

B.2.1 Error Model

Let $\mathbf{v}_0$ and $\mathbf{g}$ denote the true (teacher-generated) base velocity and guidance direction. The student produces approximations $\hat{\mathbf{v}}_0 = \mathbf{v}_0 + \boldsymbol{\epsilon}_0$ and $\hat{\mathbf{g}} = \mathbf{g} + \boldsymbol{\epsilon}_g$, where $\boldsymbol{\epsilon}_0$ and $\boldsymbol{\epsilon}_g$ are the prediction errors at timestep t. The true guided velocity is $\mathbf{v}^* = \mathbf{v}_0 + s \cdot \mathbf{g}$, and the student's approximation is $\hat{\mathbf{v}} = \hat{\mathbf{v}}_0 + s \cdot \hat{\mathbf{g}}$.

B.2.2 Per-Step Error Bound

Proposition 2 (Per-step velocity error)**.** *The velocity error at any timestep t satisfies*

$$\|\hat{\mathbf{v}}(t) - \mathbf{v}^*(t)\| = \|\boldsymbol{\epsilon}_0(t) + s \cdot \boldsymbol{\epsilon}_g(t)\| \leq \|\boldsymbol{\epsilon}_0(t)\| + s\,\|\boldsymbol{\epsilon}_g(t)\|. \tag{14}$$

Proof. Direct computation: $\hat{\mathbf{v}} - \mathbf{v}^* = (\mathbf{v}_0 + \boldsymbol{\epsilon}_0 + s(\mathbf{g} + \boldsymbol{\epsilon}_g)) - (\mathbf{v}_0 + s\mathbf{g}) = \boldsymbol{\epsilon}_0 + s\,\boldsymbol{\epsilon}_g$. The bound follows from the triangle inequality. □

At $s = 1$, both error terms contribute equally. At $s = 8$, the guidance-error contribution is amplified by a factor of 8 while the base error remains at unit weight. When guidance error dominates the base-error term, the total trajectory error approaches an $8\times$ increase over the $s = 1$ case. This asymmetry is the fundamental source of quality degradation at high guidance scales.

B.2.3 Trajectory Error Accumulation

Proposition 3 (Trajectory error over N steps)**.** *Consider an Euler discretization of the ODE $d\mathbf{x}/dt = \mathbf{v}(\mathbf{x}, t)$ with uniform step size $h = 1/N$. Let $\mathbf{x}_n$ and $\hat{\mathbf{x}}_n$ denote the true and approximate trajectories at step n. If the velocity field is Lipschitz with constant L and the per-step velocity error is bounded by $\|\hat{\mathbf{v}}(t_n) - \mathbf{v}^*(t_n)\| \leq E(s)$ for all n, then*

$$\|\hat{\mathbf{x}}_N - \mathbf{x}_N\| \leq \frac{E(s)}{L}\left(e^L - 1\right), \tag{15}$$

where $E(s) \leq \bar{\epsilon}_0 + s\,\bar{\epsilon}_g$ with $\bar{\epsilon}_0 = \max_t \|\boldsymbol{\epsilon}_0(t)\|$ and $\bar{\epsilon}_g = \max_t \|\boldsymbol{\epsilon}_g(t)\|$.

Proof. The trajectory bound is a standard result from ODE perturbation theory (Gronwall's inequality). At step $n+1$:

$$\hat{\mathbf{x}}_{n+1} - \mathbf{x}_{n+1} = (\hat{\mathbf{x}}_n - \mathbf{x}_n) + h\left(\hat{\mathbf{v}}(\hat{\mathbf{x}}_n, t_n) - \mathbf{v}^*(\mathbf{x}_n, t_n)\right). \tag{16}$$

By the Lipschitz condition and the triangle inequality:

$$\|\hat{\mathbf{x}}_{n+1} - \mathbf{x}_{n+1}\| \leq (1 + hL)\|\hat{\mathbf{x}}_n - \mathbf{x}_n\| + h\,E(s). \tag{17}$$

Applying the discrete Gronwall inequality over N steps with $h = 1/N$ yields the bound. □

The trajectory error scales linearly with $E(s) = O(\bar{\epsilon}_0 + s\,\bar{\epsilon}_g)$. When guidance error dominates ($\bar{\epsilon}_g \gg \bar{\epsilon}_0$), the trajectory error at $s = 8$ approaches $8\times$ the error at $s = 1$.

B.2.4 How Each Modulation Mechanism Reduces Amplification

Let $\gamma \triangleq w(t)\,\alpha(s)\,\varphi(a) \in (0, 1]$ denote the combined modulation factor, with $\gamma = 1$ meaning no modulation. The bounds in this subsection compare the modulated student with the equally modulated teacher field $\mathbf{v}_0 + s\gamma\,\mathbf{g}$. They therefore measure how much amplified estimation error each mechanism removes. The price of modulation, a shift of that target away from the CFG velocity $\mathbf{v}^*$, is analyzed in Section B.2.5.

Temporal Modulation $w(t)$. Replacing s with $s \cdot w(t)$ in the guided velocity bounds the effective per-step error by $\|\boldsymbol{\epsilon}_0(t)\| + s \cdot w(t) \cdot \|\boldsymbol{\epsilon}_g(t)\|$. Since $w(t) \approx 0$ at trajectory endpoints (where $\|\boldsymbol{\epsilon}_g\|$ is empirically largest due to high noise or near-convergence), the peak velocity error is reduced. The trajectory bound becomes

$$E_w(s) \leq \bar{\epsilon}_0 + s \cdot \max_t[w(t) \cdot \|\boldsymbol{\epsilon}_g(t)\|] \leq \bar{\epsilon}_0 + s \cdot \bar{\epsilon}_g^{\text{mid}}, \tag{18}$$

where $\bar{\epsilon}_g^{\text{mid}} \leq \bar{\epsilon}_g$ is the maximum guidance error in the mid-timestep window where $w(t)$ is non-negligible.

Scale Dampening $\alpha(s)$. Replacing s with $s \cdot \alpha(s)$ caps the effective amplification at $s \cdot \alpha(s) < s$ for $s > s_{\text{thresh}}$. The dampened scale yields a global reduction, giving $E_\alpha(s) \leq \bar{\epsilon}_0 + s \cdot \alpha(s) \cdot \bar{\epsilon}_g$.

Alignment Gating $\varphi(a)$. The alignment $a = \cos(\mathbf{v}_0, \mathbf{g})$ is empirically associated with guidance prediction error in our diagnostics: misaligned steps ($a < 0$) tend to coincide with larger prediction errors, likely because the guidance direction opposes the base flow and creates a harder prediction target. Applying $\varphi(a) < 1$ at these high-error timesteps reduces the effective error to $s \cdot \varphi(a) \cdot \|\boldsymbol{\epsilon}_g(t)\|$. The modulation ablation (Table 14) confirms that the gate improves FID, providing indirect evidence for the correlation. Unlike $w(t)$ (which operates in the time domain) and $\alpha(s)$ (which operates in the scale domain), $\varphi(a)$ operates in the geometric domain and adapts *per-sample* and *per-timestep* based on the actual velocity-field geometry.

Composed Modulation. Under full modulation, the student outputs $\hat{\mathbf{v}}_{\text{out}} = \hat{\mathbf{v}}_0 + s\gamma\,\hat{\mathbf{g}}$, and its estimation error against the modulated teacher field satisfies

$$\|\hat{\mathbf{v}}_{\text{out}}(t) - (\mathbf{v}_0 + s\gamma\,\mathbf{g})(t)\| \leq \|\boldsymbol{\epsilon}_0(t)\| + s \cdot w(t) \cdot \alpha(s) \cdot \varphi(a(t)) \cdot \|\boldsymbol{\epsilon}_g(t)\|. \tag{19}$$

Each factor reduces a different component of the error. Specifically, $w(t)$ suppresses temporal endpoints, $\alpha(s)$ caps the global scale, and $\varphi(a)$ targets the geometric regime where errors are largest. Because they operate on orthogonal dimensions (time, scale, geometry), their composition is multiplicative, and no single mechanism can substitute for the others. This analysis is consistent with the ablation results in Table 14, which show that every pair of mechanisms outperforms both of its components and the full trio achieves the lowest error.

B.2.5 Bias-Error Trade-off and Two-Regime Design

Equation 19 measures error against the modulated target, but the samples are ultimately judged against what CFG would produce. The following decomposition makes the gap between the two explicit.

Proposition 4 (Exact decomposition against the CFG velocity)**.** *With* $\hat{\mathbf{v}}_{out} = \hat{\mathbf{v}}_0 + s\gamma\,\hat{\mathbf{g}}$ *and* $\mathbf{v}^* = \mathbf{v}_0 + s\,\mathbf{g}$*,*

$$\hat{\mathbf{v}}_{out} - \mathbf{v}^* = \underbrace{\boldsymbol{\epsilon}_0 + s\gamma\,\boldsymbol{\epsilon}_g}_{\textit{estimation error}} + \underbrace{s(\gamma-1)\,\mathbf{g}}_{\textit{modulation bias}}, \qquad \|\hat{\mathbf{v}}_{out} - \mathbf{v}^*\| \le \|\boldsymbol{\epsilon}_0\| + s\gamma\,\|\boldsymbol{\epsilon}_g\| + s(1-\gamma)\,\|\mathbf{g}\|. \tag{20}$$

Proof. Substituting $\hat{\mathbf{v}}_0 = \mathbf{v}_0 + \boldsymbol{\epsilon}_0$ and $\hat{\mathbf{g}} = \mathbf{g} + \boldsymbol{\epsilon}_g$ gives $\hat{\mathbf{v}}_{\text{out}} - \mathbf{v}^* = \boldsymbol{\epsilon}_0 + s\gamma\,\boldsymbol{\epsilon}_g + s(\gamma-1)\,\mathbf{g}$. Since $\gamma \in (0,1]$, $|\gamma - 1| = 1 - \gamma$, and the bound follows from the triangle inequality. The identity holds for any value of γ, including the one computed from the student's own alignment $\hat{a} = \cos(\hat{\mathbf{v}}_0, \hat{\mathbf{g}})$ at inference. □

Interpretation. The first two terms are the estimation error, which modulation reduces by shrinking the coefficient on $\boldsymbol{\epsilon}_g$ from s to $s\gamma$. The third term is a bias, because the output now follows $\mathbf{v}_0 + s\gamma\,\mathbf{g}$ rather than the CFG velocity. The bias is not small at high scale. At $s = 8$ and the window center, where $w = \varphi = 1$, dampening alone gives $\gamma = \alpha(8) = 0.7$ and a bias of $2.4\,\|\mathbf{g}\|$. Modulation therefore pays off only if CFG's own high-scale dynamics are worth leaving, which is an empirical question.

Two Regimes. The design separates two regimes at $s_{\text{thresh}} = 4$. At or below the threshold, $\alpha(s) = 1$ and $\lambda(s) = 0$, the temporal window is broad, and $\varphi(a) = 1$ whenever $a \ge 0$, so modulation is mild and CtrlFlow tracks CFG closely (3.07 versus 2.97 at $(25, 2)$). Above the threshold, the target departs from CFG deliberately, and three measurements support this choice.

- *Zero-error control.* Exact two-pass CFG has $\boldsymbol{\epsilon}_0 = \boldsymbol{\epsilon}_g = \mathbf{0}$, so any change it shows under modulation comes from the bias term alone. Applying $w\alpha$ alone lowers its FID from 53.84 to 46.02 at $(4, 8)$ and from 21.47 to 17.89 at $(25, 8)$, and adding warping, which changes only the discretization, lowers it further to 42.31 and 17.42 (Table 17).
- *Component swap.* The unmodulated pipeline gives 21.47 FID at $(25, 8)$ with exact teacher components and 21.74 to 22.31 when one of them is replaced by the learned one, whereas CtrlFlow-Full reaches 19.28 (Table 4), so the gain cannot come from a closer CFG approximation.
- *Relative size.* On matched states at $s = 8$, the integrated bias $\int s(1-\gamma)\,\|\mathbf{g}\|\,dt$ exceeds the integrated amplified error $\int s\gamma\,\|\boldsymbol{\epsilon}_g\|\,dt$ by about $3.2\times$, so the dominant effect of modulation at high scale is a deliberate shift of the target.

Time-grid warping does not enter Equation 20, since it changes where the solver evaluates the field rather than the field itself (Section B.3). Because the high-scale target is intentionally not CFG, the CFG-Match criterion of Section 4.1 counts improvements over CFG as matches, which makes it one-sided.

B.3 Time-Warp Properties

This section proves the formal properties of the time-grid warping mechanism (Section 3.5) and discusses its relationship to discretization error.

B.3.1 Monotonicity and Endpoint Preservation

Proposition 5 (Well-definedness of τ)**.** *Let* $\omega : [0,1] \to \mathbb{R}_{>0}$ *be a continuous, strictly positive function. Define*

$$\tau(t) = \frac{\int_0^t \omega(u)\,du}{\int_0^1 \omega(u)\,du}. \tag{21}$$

Then τ *is a valid time-warping function. It is strictly monotonically increasing, continuously differentiable, and satisfies* $\tau(0) = 0$ *and* $\tau(1) = 1$*.*

Proof. Endpoint preservation. $\tau(0) = 0/\int_0^1 \omega(u)\,du = 0$ and $\tau(1) = \int_0^1 \omega(u)\,du/\int_0^1 \omega(u)\,du = 1$.

Monotonicity. The derivative is $\tau'(t) = \omega(t)/\int_0^1 \omega(u)\,du$. Since $\omega(t) > 0$ for all $t \in [0,1]$ and the denominator is a positive constant, $\tau'(t) > 0$ everywhere. Hence τ is strictly increasing.

Smoothness. Since ω is continuous, τ is continuously differentiable by the fundamental theorem of calculus. □

Corollary 6. *The CtrlFlow dilation weight $\omega(t) = 1 + \lambda(s) \cdot d(t)$ satisfies $\omega(t) > 0$ for all t and all $s \geq 0$, since $\lambda(s) \geq 0$ and $d(t) = \mathrm{sigmoid}(\cdot) \in (0,1)$. Hence the resulting τ is always a valid warping function.*

Remark 1 (No velocity-field bias). Because τ is strictly increasing with $\tau(0) = 0$ and $\tau(1) = 1$, warping only moves the N solver timesteps along the same trajectory interval and leaves the velocity field being integrated unchanged. It therefore adds no term to the bias-error decomposition of Proposition 4, and it affects only the discretization error analyzed below.

B.3.2 Discretization Error Reduction

The following proposition formalizes the intuition that concentrating solver evaluations near high-curvature regions reduces ODE integration error.

Proposition 7 (Curvature-weighted step allocation)**.** *Consider an ODE $d\mathbf{x}/dt = \mathbf{v}(\mathbf{x}, t)$ integrated with Euler's method using step sizes $h_n = t_{n+1} - t_n$. The local truncation error at step n is*

$$\boldsymbol{e}_n = \frac{h_n^2}{2}\left.\frac{d^2\mathbf{x}}{dt^2}\right|_{t=t_n} + O(h_n^3) = \frac{h_n^2}{2}\left(\frac{\partial \mathbf{v}}{\partial t} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\,\mathbf{v}\right)\Bigg|_{t_n} + O(h_n^3). \tag{22}$$

Let $c_n = \|d^2\mathbf{x}/dt^2|_{t_n}\|$. For a fixed total number of steps N (so $\sum_n h_n = 1$), reducing h_n in regions with larger c_n decreases the leading-order error. The exact optimal exponent depends on the aggregate error objective. Minimizing $\sum_n h_n^2 c_n$ subject to $\sum_n h_n = 1$ gives $h_n \propto c_n^{-1}$, while equidistributing local error magnitudes $h_n^2 c_n$ gives $h_n \propto c_n^{-1/2}$. Both allocations place smaller steps where curvature is larger.

Proof. The local truncation error follows from Taylor expansion of the exact solution. The total error is $E = \sum_{n=0}^{N-1} \|\boldsymbol{e}_n\| \approx \frac{1}{2}\sum_n h_n^2\, c_n$ where $c_n = \|d^2\mathbf{x}/dt^2|_{t_n}\|$. Minimizing $E = \sum_n h_n^2 c_n$ subject to $\sum_n h_n = 1$ via Lagrange multipliers gives the first-order condition $h_n c_n = \text{const}$, yielding $h_n \propto c_n^{-1}$. Alternatively, equidistributing local error $h_n^2 c_n = \text{const}$ gives $h_n \propto c_n^{-1/2}$. Both allocations assign smaller steps to regions with larger trajectory curvature. □

Connection to CtrlFlow's Warping. The second derivative of the guided trajectory $\|d^2\mathbf{x}/dt^2\|$ depends on the velocity field's spatial and temporal curvature. At high guidance scales, the curvature concentrates in the mid-timestep region where $\|\mathbf{g}\|$ is large and changes rapidly. At trajectory endpoints, the velocity field is smoother, being nearly unconditional at $t \approx 1$ and nearly converged at $t \approx 0$.

CtrlFlow's schedule map τ approximates a curvature-weighted step allocation. The dilation weight $\omega(t)$ is large at trajectory endpoints, so the schedule map advances quickly there and the solver takes wider steps. In the mid-timestep region, $\omega(t)$ is small, so the schedule map advances slowly and the solver takes finer steps. The resulting grid concentrates evaluations where the velocity field curvature is highest. The warping is not strictly optimal in the sense of Proposition 7, since it uses a heuristic $d(t)$ rather than the true second derivative. It nonetheless aligns with the qualitative structure of curvature-weighted allocation and empirically reduces the dead-zone phenomenon (Figure 4). The Euler derivation explains why smaller local steps should be allocated to high-curvature regions. DPM-Solver has different truncation constants, yet the same qualitative dependence on schedule allocation motivates the heuristic.

B.3.3 Computational Cost

The warped schedule $\{t_0, t_1, \ldots, t_N\}$ is computed by evaluating the CDF-like integral $\tau(i/N)$ at $N+1$ points. The trapezoidal rule on a fine grid of K points yields the following costs.

- Evaluate $\omega(t_k)$ at K points, requiring $O(K)$ operations. Each evaluation computes $m(t_k)$, $d(t_k)$, and $\omega(t_k)$, all closed-form expressions.
- Compute the cumulative sum $\int_0^{t_k} \omega(u)\,du$ via prefix sum in $O(K)$.
- Normalize by $\int_0^1 \omega(u)\,du$ in $O(K)$.
- Look up $\tau(i/N)$ by interpolation in $O(N \log K)$ with binary search.

With $K = 1000$ (our default), the total precomputation is negligible (< 1 ms) relative to the forward-pass cost (≈ 17 ms per step). The warped schedule is computed once per (N, s) configuration and cached.

C Hyperparameter Specification

This appendix lists the functional forms and numerical values of every hyperparameter introduced in the methodology.

C.1 Modulation Functions (Section 3.4)

Temporal Window $w(t)$. The temporal modulation is a Gaussian window,

$$w(t) = \exp\left(-\frac{(t-\mu)^2}{2\,\sigma(s)^2}\right), \tag{23}$$

where μ centers the window and $\sigma(s)$ controls its width.

Temporal Window Width $\sigma(s)$. The width of the Gaussian temporal modulation $w(t)$ (Equation 23) decreases linearly with s, following

$$\sigma(s) = \sigma_{\max} - (\sigma_{\max} - \sigma_{\min}) \cdot \frac{\min(s, s_{\max}) - 1}{s_{\max} - 1}, \tag{24}$$

where $\sigma_{\max} = 0.35$ produces a broad window at $s = 1$ and $\sigma_{\min} = 0.10$ yields a narrow window at $s = s_{\max}$. The window center remains fixed at $\mu = 0.5$, the midpoint of the $[0, 1]$ time interval.

Scale Dampening $\alpha_{\text{damp}}(s)$. The dampening function (Section 3.4) applies a linear decay above the activation threshold, given by

$$\alpha_{\text{damp}}(s) = 1 - \beta \cdot \frac{s - s_{\text{thresh}}}{s_{\max} - s_{\text{thresh}}}, \tag{25}$$

where $\beta = 0.3$ controls the maximum dampening. At $s = s_{\max} = 8$, $\alpha_{\text{damp}} = 0.7$, reducing the effective guidance scale from 8 to 5.6. Exponential and sigmoid alternatives were also tested. Linear decay matched their performance while remaining simpler to implement.

Alignment Gating $\varphi(a)$. As defined in Section 3.4, $\varphi(a) = 1$ for $a \geq 0$ and $\varphi(a) = \max(\varphi_{\min}, 1 + a)$ for $a < 0$, with $\varphi_{\min} = 0.1$.

C.2 Time-Warp Parameters (Section 3.5)

Time Score Parameters. The time score $m(t) = \exp(-(t - \mu_w)^2/(2\sigma_w^2))$ uses $\mu_w = 0.5$ and $\sigma_w = 0.25$. These values are chosen independently of the temporal modulation parameters μ and $\sigma(s)$. This separation lets the warping respond to a broader region of the trajectory.

Deadness Gate. The soft gate $d(t) = \text{sigmoid}(\kappa \cdot (m_0 - m(t)))$ uses threshold $m_0 = 0.3$ and sharpness $\kappa = 10$. With $m_0 = 0.3$, the gate transitions near $t = 0.15$ and $t = 0.85$, flagging the outer 30% of the trajectory as dead-zone-prone.

Warp Magnitude. $\lambda(s) = \lambda_0 \cdot \max(0, s - s_{\text{thresh}})/s_{\max}$ with $\lambda_0 = 1.0$. At $s = 8$, $\lambda(8) = 1.0 \cdot (8 - 4)/8 = 0.5$. The maximum dilation weight therefore reaches $\omega_{\max} = 1 + \lambda(s) \cdot 1 = 1.5$, because $d(t) \leq 1$.

C.3 Shared Parameters

The activation threshold $s_{\text{thresh}} = 4$ is shared by the scale dampening $\alpha(s)$ and the time-warp magnitude $\lambda(s)$. Below this threshold, both mechanisms remain inactive, leaving low-s generation unmodified.

The training range upper bound $s_{\max} = 8$ normalizes $\sigma(s)$, $\alpha_{\text{damp}}(s)$, and $\lambda(s)$.

C.4 Summary Table

Table 5: Complete hyperparameter specification for CtrlFlow's modulation and warping mechanisms.

Symbol	Description	Value	Used in
Temporal Modulation			
μ	Window center	0.5	Eq. 23
$\sigma_{\max}$	Window width at s=1	0.35	$\sigma(s)$
$\sigma_{\min}$	Window width at s=$s_{\max}$	0.10	$\sigma(s)$
Scale Dampening			
s_{thresh}	Activation threshold	4.0	Sec. 3.4
β	Maximum dampening	0.3	$\alpha_{\text{damp}}(s)$
Alignment Gating			
$\varphi_{\min}$	Minimum gate value	0.1	Sec. 3.4
Time-Grid Warping			
μ_w	Time score center	0.5	$m(t)$
σ_w	Time score width	0.25	$m(t)$
m_0	Deadness threshold	0.3	$d(t)$
κ	Gate sharpness	10	$d(t)$
λ_0	Base warp magnitude	1.0	$\lambda(s)$
Shared			
s_{thresh}	High-s activation	4.0	$\alpha(s)$, $\lambda(s)$
$s_{\max}$	Training range upper bound	8.0	$\sigma(s)$, $\alpha_{\text{damp}}(s)$, $\lambda(s)$
Architecture			
r	LoRA rank	64	Eq. 5
Training			
s_{target}	Phase 1 fixed scale	2.0	Phase 1
Phase 1 iterations		15K	
Phase 1 learning rate		5×10^{-5}	
Phase 2 iterations		20K	
Phase 2 learning rate		3×10^{-5}	
Phase 3 iterations		13K	
Phase 3 learning rate		1×10^{-5}	
Batch size		256	
Optimizer	AdamW	β_1=0.9, β_2=0.999, wd 10^{-2}	

C.5 Sensitivity Discussion

The modulation hyperparameters (μ, $\sigma_{\max}$, $\sigma_{\min}$, s_{thresh}, β, $\varphi_{\min}$) were tuned on a validation set of 5K images across five (N, s) configurations. These configurations span $(25, 2)$, $(25, 8)$, $(8, 4)$, $(4, 2)$, and $(4, 8)$. Among all parameters, s_{thresh} proved the most sensitive. Values below 3 degraded diversity at moderate scales, while values above 5 left high-s artifacts uncorrected. The remaining parameters tolerate $\pm 20\%$ perturbations from their reported values, with FID shifts below 0.5 on the validation configurations. Because these five configurations coincide with the headline configurations of Table 2, we check robustness on configurations never used for selection. CtrlFlow matches CFG on all 20 untuned grid configurations except $(2, 8)$ (Table 6), and the mechanism-transfer test (Table 17) includes the untuned $(16, 6)$ and $(8, 6)$, where the same ordering holds with all hyperparameters frozen at the values above. A full grid search over all hyperparameters on additional backbones is left to future work.

D Extended Experimental Results

This appendix presents supplementary experiments that reinforce the main-text findings.

D.1 Full Configuration Grid

Table 2 in the main text reports five representative configurations restricted to FID. Table 6 expands this to the complete 25-configuration grid for the fixed-checkpoint comparison among CFG, CtrlFlow, and the fixed-scale AGD variant, validating the 96% CFG-match accuracy claim. The matched scale-conditioned baseline appears in Appendix D.2. The only configuration where CtrlFlow exceeds the match threshold δ is $(N{=}2, s{=}8)$. In our protocol, two solver steps under-resolve the ODE trajectory even after warping.

Table 6: Full (N, s) grid. FID-50K (↓) for all 25 configurations on ImageNet 256×256. The boxed cell marks the sole configuration exceeding the CFG-match threshold δ. Values in **bold** indicate CtrlFlow outperforms CFG at that configuration.

Method		$N{=}2$	$N{=}4$	$N{=}8$	$N{=}16$	$N{=}25$
CFG	$s{=}1$	42.51	21.83	13.45	10.21	9.62
	$s{=}2$	21.73	9.24	4.82	3.45	2.97
	$s{=}4$	38.54	19.67	11.72	8.63	7.83
	$s{=}6$	64.28	36.18	20.47	16.02	14.21
	$s{=}8$	86.53	53.84	33.18	24.38	21.47
CtrlFlow	$s{=}1$	43.89	22.17	13.82	10.53	9.87
	$s{=}2$	22.18	9.47	4.96	3.52	3.07
	$s{=}4$	**37.42**	**18.84**	**10.98**	**8.47**	**7.51**
	$s{=}6$	**62.83**	**33.05**	**19.21**	**14.38**	**12.68**
	$s{=}8$	97.34	**48.95**	**30.12**	**22.14**	**19.28**
AGD (fixed s)	$s{=}1$	50.51	27.33	17.25	13.41	12.34
	$s{=}2$	22.87	10.08	5.31	3.63	3.04
	$s{=}4$	63.54	38.17	25.81	19.13	16.72
	$s{=}6$	98.28	63.18	41.47	33.52	28.93
	$s{=}8$	121.53	82.17	63.82	50.38	44.92

D.2 Scale-Conditioned Opaque Baseline

The fixed-scale comparison above tests single-checkpoint behavior outside a training scale, which makes its high-scale failure predictable. The more informative baseline for judging the decomposition is a student that sees the same range of scales but keeps a single opaque output. AGD-MultiScale is that student. It follows the AGD design [Jensen and Sadat, 2025], receiving s as a network input embedded in the same way as the timestep, and it is trained with $s \sim \mathcal{U}[1, 8]$ under the same three-phase curriculum, teacher, trajectories, and optimizer as CtrlFlow, at matched capacity (rank 128, 34M parameters). It is therefore the vanilla scale-conditioned guidance-distillation baseline in the sense of Meng et al. [2023]. The two students share capacity, teacher, data, and curriculum and differ in their output interface: AGD-MultiScale receives s as an input and returns one guided velocity, whereas CtrlFlow returns $\mathbf{v}_0$ and $\mathbf{g}$ and applies s afterwards.

Table 7: Matched comparison with the scale-conditioned opaque student, FID-50K (↓) under the protocol of Table 2. "Grid match" counts configurations with $\mathrm{FID} \leq \mathrm{FID}_{\mathrm{CFG}} + \delta$ over all 25 cells.

Method	Scale input	Params	$(25, 2)$	$(8, 4)$	$(25, 8)$	$(4, 8)$	Grid match
AGD (fixed $s{=}2$)	none	14M	3.04	25.81	44.92	82.17	5/25
AGD-MultiScale ($r{=}128$)	before forward	34M	3.09	11.94	23.42	52.87	20/25
CtrlFlow-Full	after forward	34M	3.07	**10.98**	**19.28**	**48.95**	**24/25**

With equal parameters, training data, and supervision, CtrlFlow is on par at $(25, 2)$, where the 0.02 FID gap is below seed variation, and lower elsewhere, with the margin growing where guidance errors are amplified, to 4.14 at $(25, 8)$ and 3.92 at $(4, 8)$. The five grid misses of AGD-MultiScale concentrate at low N and high s. What the two students do not share is exactly the interface: post-forward composition of s, per-step access to $\|\mathbf{g}\|$ and $\cos(\mathbf{v}_0, \mathbf{g})$, and the alignment gate that uses

them. Appendix D.12 tests how much of the gap remains once the opaque student also receives every mechanism that does not need the split.

D.3 Precision and Recall

Table 2 in the main text is limited to FID. Table 8 supplements it with Precision and Recall at six representative (N, s) configurations for the three single-pass-relevant methods. The most notable result appears at $(25, 8)$, where the Recall of the fixed-scale AGD variant collapses to 0.14 compared with CFG's 0.32, signaling severe mode dropping. CtrlFlow sustains Recall at 0.35 while preserving Precision at 0.86. The gap widens further at $(4, 8)$, where its Recall falls to 0.06 versus CtrlFlow's 0.24.

Table 8: Precision (↑) / Recall (↑) at six representative (N, s) configurations. Best single-pass result per column in **bold**.

Method	$(25, 2)$	$(25, 8)$	$(8, 2)$	$(8, 8)$	$(4, 2)$	$(4, 8)$
CFG	.82 / .57	.87 / .32	.79 / .55	.84 / .28	.73 / .52	.78 / .22
AGD (fixed s)	.81 / .55	.72 / .14	.78 / .53	.65 / .09	.71 / .50	.58 / .06
CtrlFlow	**.82 / .56**	**.86 / .35**	**.79 / .54**	**.83 / .30**	**.73 / .51**	**.76 / .24**

D.4 Inception Score

Table 9 reports Inception Score (IS) at six representative configurations, including the main easy and high-scale regimes. IS provides a complementary quality signal to FID by measuring both per-sample classifiability and overall diversity.

Table 9: Inception Score (↑) at six representative (N, s) configurations. Best single-pass result per column in **bold**.

Method	$(25, 2)$	$(25, 8)$	$(8, 2)$	$(8, 8)$	$(4, 2)$	$(4, 8)$
CFG	271.3	198.4	248.6	167.2	201.5	108.3
AGD (fixed s)	264.8	112.7	241.3	82.4	193.7	47.6
CtrlFlow	**269.5**	**201.2**	**246.1**	**170.8**	**198.3**	**112.7**

D.5 Class Adherence

FID measures how close the sample distribution is to real images, but it does not check whether each sample shows the requested class. This check matters for CtrlFlow because temporal windowing and scale dampening reduce the effective guidance at high s, which could in principle lower FID by weakening conditioning. Table 10 reports the top-1 and top-5 accuracy of a frozen ResNet-50 classifier [He et al., 2015] (torchvision IMAGENET1K_V2 weights) with respect to the conditioning class, computed on the same 50K-sample seed lists used for FID. A ConvNeXt-B classifier agrees within 0.4 points.

Table 10: Class adherence. Top-1 / top-5 accuracy (%, ↑) of a frozen ResNet-50 on generated samples with respect to the conditioning class.

Method	$(25, 2)$	$(16, 6)$	$(25, 8)$	$(4, 8)$
CFG	90.1 / 98.4	87.3 / 97.2	85.6 / 96.5	77.8 / 93.1
AGD-MultiScale	89.6 / 98.1	85.8 / 96.5	79.4 / 93.8	57.3 / 82.4
CtrlFlow-Core	89.4 / 98.0	85.1 / 96.2	83.7 / 95.6	69.2 / 89.1
CtrlFlow-Full	89.9 / 98.3	86.9 / 97.0	85.1 / 96.3	75.9 / 92.2

CtrlFlow-Full stays within 0.5 points of CFG's top-1 accuracy at $(25, 2)$, $(16, 6)$, and $(25, 8)$, and within 1.9 points at $(4, 8)$, where $\alpha(8) = 0.7$ is fully engaged. The FID gains therefore reflect a favorable balance between image quality and condition alignment rather than silently weakened conditioning. Modulation also recovers most of the adherence that Core loses at high scale (75.9%

against 69.2% at $(4, 8)$), whereas the opaque student falls to 57.3%. Inception Score in Table 9 points the same way.

D.6 FID Standard Deviations

All 25 CtrlFlow configurations were evaluated over 3 generation seeds. All baselines share the same three seed lists, so every entry in Table 2 is a 3-seed mean. Table 11 reports the standard deviation of FID-50K across those seeds for CtrlFlow, whose CFG-match claim depends on per-configuration stability. The 24/25 CFG-match classification remains stable under all seeds, with no near-threshold configuration switching between match and non-match status.

Table 11: FID-50K standard deviation ($\pm$) across 3 generation seeds for CtrlFlow on DiT-XL/2.

	N=2	N=4	N=8	N=16	N=25
s=1	$\pm$0.41	$\pm$0.23	$\pm$0.19	$\pm$0.15	$\pm$0.11
s=2	$\pm$0.26	$\pm$0.16	$\pm$0.10	$\pm$0.09	$\pm$0.07
s=4	$\pm$0.39	$\pm$0.24	$\pm$0.18	$\pm$0.14	$\pm$0.12
s=6	$\pm$0.55	$\pm$0.34	$\pm$0.26	$\pm$0.21	$\pm$0.18
s=8	$\pm$0.94	$\pm$0.51	$\pm$0.37	$\pm$0.30	$\pm$0.26

D.7 Training Convergence

Table 12 reports validation FID at two representative configurations across the three curriculum phases. During Phase 1, the model rapidly establishes a low-s baseline. FID at $(25, 2)$ drops from 45.2 to 3.41 within 15K iterations, whereas $(25, 8)$ stays above 80 because the model has not yet encountered high-s targets. Phase 2 introduces variable s, causing FID at $(25, 8)$ to fall sharply from 82.3 to 21.8 over 20K iterations. Phase 3 refines performance across diverse (N, s) pairs, bringing both configurations to their final values. The curriculum ablation in Table 15 confirms that removing any phase degrades the final model.

Table 12: Training convergence. Validation FID-50K ($\downarrow$) at two configurations measured at the end of each curriculum phase.

Configuration	End of Phase 1	End of Phase 2	End of Phase 3
$(25, 2)$	3.41	3.24	3.07
$(25, 8)$	82.3	21.8	19.28

D.8 Component Ablation

Table 13 removes each mechanism individually from the full system. Unified distillation produces the largest effect. Replacing it with a naive dual-loss degrades FID by 2.3 at $(4, 2)$ and by 19.5 at $(4, 8)$. Guidance modulation and time warping contribute modestly at moderate (N, s) yet become essential at extreme settings. Removing modulation at $(4, 8)$ raises FID from 48.95 to 59.83, while removing time warping pushes it to 63.42.

Table 13: Component ablation. FID-50K ($\downarrow$) at four (N, s) configurations. Each row removes one mechanism while keeping the others active.

Variant	$(4, 2)$	$(4, 8)$	$(16, 2)$	$(16, 8)$
Full CtrlFlow	9.47	48.95	3.52	22.14
w/o unified distillation	11.73	68.41	4.31	32.87
w/o guidance modulation	9.54	59.83	3.57	28.91
w/o time warping	9.51	63.42	3.54	24.67

D.9 Per-Modulation Sub-Ablation

The component ablation treats guidance modulation as a single block. Table 14 isolates the three sub-components at $(4, 8)$ and $(16, 8)$, the two configurations where modulation matters most. Each sub-component alone yields partial improvement over no modulation. Every pair outperforms both of its components, and the full trio achieves the lowest FID at both configurations. The alignment gate $\varphi(a)$ alone produces the smallest individual gain (56.89 vs. 59.83 at $(4, 8)$). Combining $w(t) \cdot \varphi(a)$ reaches 52.18, improving 1.3 FID over $w(t)$ alone (53.47) and 4.7 FID over $\varphi(a)$ alone (56.89). This gap supports the claim that the mechanisms are complementary rather than redundant.

Table 14: Per-modulation sub-ablation. FID-50K (↓). Each row activates only the listed sub-components.

Active sub-components	$(4, 8)$	$(16, 8)$
$w(t) \cdot \alpha(s) \cdot \varphi(a)$ (full)	**48.95**	**22.14**
$w(t) \cdot \alpha(s)$	51.34	23.87
$w(t) \cdot \varphi(a)$	52.18	24.15
$\alpha(s) \cdot \varphi(a)$	54.63	25.94
$w(t)$ only	53.47	25.31
$\alpha(s)$ only	55.21	26.83
$\varphi(a)$ only	56.89	27.42
None	59.83	28.91

D.10 Curriculum Ablation

Table 15 evaluates the three-phase training strategy. Skipping Phase 1 (direct multi-scale training) destabilizes convergence and opens a 13.1 FID gap at $(16, 8)$. Skipping Phase 2 (no scale variation before step-count variation) causes overfitting to the fixed scale, degrading high-s performance. Training all (N, s) jointly from the start converges 1.7× slower and produces worse final FID than the phased schedule.

Table 15: Curriculum ablation. FID-50K (↓) at two configurations.

Variant	FID $(16, 2)$	FID $(16, 8)$	Converged?
Full 3-phase	3.52	22.14	Yes
Skip Phase 1	5.21	35.28	Unstable
Skip Phase 2	3.74	37.93	Overfits
All (N, s) from start	4.91	28.64	Slow

D.11 LoRA Rank Sensitivity

Table 16 varies the LoRA rank r while holding all other hyperparameters fixed. The easy configuration $(25, 2)$ is largely insensitive to rank, showing only a 0.5 FID gap between r=16 and r=128. Harder configurations expose the capacity bottleneck. At $(25, 8)$, halving the rank from 64 to 32 increases FID by 1.59 (8.2%), and at $(4, 8)$ the increase reaches 4.51 (9.2%). Doubling the rank from 64 to 128 yields less than 2% improvement at any configuration, despite doubling the adapter parameter count. Rank 64 therefore sits at the knee of the diminishing-returns curve.

Table 16: LoRA rank sensitivity. FID-50K (↓) and parameter overhead at three configurations. Rank 64 sits at the knee of the cost-quality curve.

Rank r	Extra params	FID $(25, 2)$	FID $(25, 8)$	FID $(4, 8)$
16	1.3%	3.54	24.18	61.34
32	2.5%	3.21	20.87	53.46
64 (default)	5.0%	**3.07**	**19.28**	**48.95**
128	10.0%	3.03	19.02	48.17

D.12 Mechanism Transfer to Baselines

The ablations above show that modulation and warping matter, but not whether their benefit requires the decomposition. The three mechanisms differ in this respect. Temporal windowing $w(t)$ and dampening $\alpha(s)$ only change the effective scale, so any method that exposes a scale can use them. Time warping acts on the solver and therefore transfers to every method. Only the alignment gate $\varphi(a)$ needs $\mathbf{v}_0$ and $\mathbf{g}$ at every step, which two-pass CFG supplies at $2N$ evaluations, CtrlFlow supplies at N, and no opaque single-pass student supplies at any cost. For the opaque student, the faithful analogue of $w \cdot \alpha$ is a time-varying scale input $s_{\text{eff}}(t) = s\, w(t)\, \alpha(s)$, because multiplying its output would also rescale the base flow. All hyperparameters stay frozen at the values of Appendix C, and two of the four test configurations, $(16, 6)$ and $(8, 6)$, were never used for selection.

Table 17: Mechanism transfer, FID-50K (↓) on shared seed lists. Bold marks the best single-pass (NFE $= N$) entry per column. $(16, 6)$ and $(8, 6)$ were never used for hyperparameter selection.

Variant	NFE	$(4, 8)$	$(25, 8)$	$(16, 6)$	$(8, 6)$
CFG	$2N$	53.84	21.47	16.02	20.47
CFG + $w\alpha$	$2N$	46.02	17.89	15.41	19.58
CFG + warp	$2N$	48.51	21.04	15.63	19.82
CFG + $w\alpha$ + warp	$2N$	42.31	17.42	14.88	19.04
CFG + $w\alpha\varphi$ + warp	$2N$	41.03	16.31	14.22	18.51
AGD-MultiScale	N	52.87	23.42	16.90	21.40
AGD-MultiScale + $s_{\text{eff}}(t)$ + warp	N	50.60	21.40	16.20	20.50
CtrlFlow-Core	N	73.41	23.65	16.10	21.00
CtrlFlow-Core + $w\alpha$ + warp	N	51.34	20.90	15.00	20.20
CtrlFlow-Full	N	**48.95**	**19.28**	**14.38**	**19.21**

Three contrasts in Table 17 separate the sources of gain. Comparing Full with Core plus $w\alpha$ and warping isolates the alignment gate, which is worth 0.6 to 2.4 FID and matters most at high s. Comparing Full with AGD-MultiScale plus $s_{\text{eff}}(t)$ and warping isolates the interface once the opaque student has every transferable mechanism, and Full remains 1.3 to 2.1 FID lower at all four configurations, including the two untuned ones. The transferable mechanisms do help two-pass CFG and the opaque student, by up to 11.5 and 2.3 FID respectively, so they are not specific to the decomposition. They do not, however, close the single-pass gap without the interface. Finally, two-pass CFG with all mechanisms sets a ceiling. CtrlFlow-Full comes within 0.16 FID of it at $(16, 6)$ and 0.70 at $(8, 6)$ at half the evaluations, while the gap widens to 2.97 at $(25, 8)$ and 7.92 at $(4, 8)$, which measures what the second evaluation still buys in the hardest regimes.

D.13 Dead-Zone Analysis

Figure 4 displays the FID ratio between CtrlFlow and CFG across the full (N, s) grid, both with and without time-grid warping. Without warping, the ratio exceeds 1.10 at four configurations where $N \leq 4$ and $s \geq 6$, corroborating the dead-zone phenomenon. With warping enabled, all but one configuration remain at or below 1.04. The sole exception is $(N{=}2, s{=}8)$, where two solver steps appear insufficient to resolve the ODE trajectory even after warping (ratio 1.12). Even at that extreme, warping reduces the ratio from 1.35 to 1.12, a 23-percentage-point improvement.

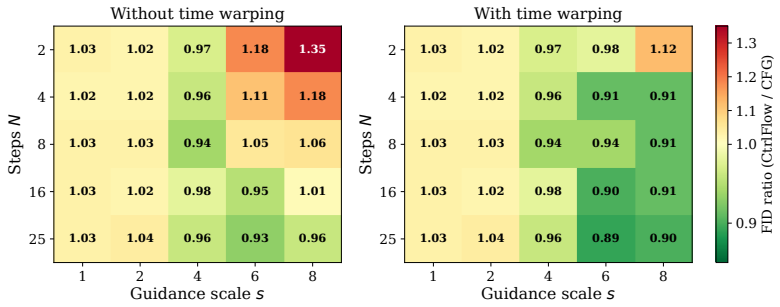

Figure 4: FID ratio (CtrlFlow / CFG) across the (N, s) grid. **Left.** Without time warping, dead zones appear at low N and high s (ratio > 1.10, four configurations at $s \geq 6$). At $s \leq 4$, the warping magnitude is zero and both panels are identical. **Right.** With time warping, all configurations except $(2, 8)$ stay at or below 1.04. High-s configurations improve over CFG (green cells, ratio < 1).

D.14 Scale Extrapolation

CtrlFlow exposes s as a post-hoc inference parameter, although training covers only $s \in [1, 8]$. Table 18 tests $s \in \{10, 12\}$ beyond the training range at N=25. CtrlFlow degrades gracefully, with FID increasing from 19.28 at s=8 to 28.41 at s=10 and 45.92 at s=12. At s=10, CtrlFlow still outperforms CFG (28.41 vs. 31.24) because the modulation mechanisms dampen the error amplification that also affects CFG at extreme scales. At s=12, CtrlFlow trails CFG by 3.1 FID, a moderate gap considering this scale lies 50% beyond the training boundary. The fixed-scale AGD variant, by contrast, degrades sharply at both extrapolated scales (FID > 60).

Table 18: Scale extrapolation beyond training range ($s_{\max}$=8). FID-50K (↓) at N=25.

Method	$s = 8$	$s = 10$	$s = 12$
CFG	21.47	31.24	42.81
AGD (fixed s)	44.92	62.38	78.14
CtrlFlow	**19.28**	**28.41**	45.92

D.15 Cross-Backbone Generalization

All main-text experiments use DiT-XL/2 (675M parameters, patch size 2). To assess whether the decomposition transfers to a different architecture scale, we train CtrlFlow on DiT-L/4 (458M parameters, patch size 4) using an identical protocol. Each LoRA adapter adds 2.5% of DiT-L/4's parameters (rank r=64). Table 19 reports FID at four representative (N, s) configurations.

Table 19: Cross-backbone generalization. FID-50K (↓) on DiT-L/4 (458M) at four (N, s) configurations. CtrlFlow uses the same training protocol as on DiT-XL/2.

Method	$(25, 2)$	$(25, 8)$	$(8, 4)$	$(4, 8)$
CFG (DiT-L/4)	5.52	34.82	20.14	82.41
AGD, fixed s (DiT-L/4)	5.78	72.34	42.18	118.52
CtrlFlow (DiT-L/4)	5.71	32.14	19.23	76.38
DiT-XL/2 reference (from Table 2)				
CFG (DiT-XL/2)	2.97	21.47	11.72	53.84
CtrlFlow (DiT-XL/2)	3.07	19.28	10.98	48.95

On DiT-L/4, CtrlFlow stays within 0.19 FID of CFG at $(25, 2)$ and surpasses CFG at high scales. CtrlFlow reaches 32.14 vs. 34.82 at $(25, 8)$ and 76.38 vs. 82.41 at $(4, 8)$. The fixed-scale AGD variant collapses at high s on DiT-L/4 with the same severity observed on DiT-XL/2 (FID 72.34 at $(25, 8)$ vs. CFG's 34.82). The CFG-match gap profile remains consistent across backbones, supporting the claim that the decomposition transfers to DiT-L/4 under our protocol.

Table 20 reports decomposition fidelity on DiT-L/4. Both components peak at t=0.5 and dip at trajectory endpoints, following the same temporal pattern as DiT-XL/2. Absolute cosine similarities are 0.01 to 0.025 lower than on DiT-XL/2, consistent with the smaller backbone offering a less expressive feature space.

Table 20: Decomposition fidelity on DiT-L/4. Cosine similarity (↑) between learned and teacher-generated components, averaged over 1K samples at s=2.

Component	t=0.8	t=0.5	t=0.2	t=0.05
$\cos(\hat{\mathbf{v}}_0,\, \mathbf{v}_0^*)$ (DiT-L/4)	0.931	0.958	0.946	0.898
$\cos(\hat{\mathbf{g}},\, \mathbf{g}^*)$ (DiT-L/4)	0.812	0.907	0.889	0.843
DiT-XL/2 reference (from Table 23)				
$\cos(\hat{\mathbf{v}}_0,\, \mathbf{v}_0^*)$ (DiT-XL/2)	0.942	0.968	0.957	0.911
$\cos(\hat{\mathbf{g}},\, \mathbf{g}^*)$ (DiT-XL/2)	0.836	0.924	0.908	0.867

D.16 Comparison with Guidance-Free Training

GFT [Chen et al., 2025a] offers single-pass sampling with a guidance temperature β_{GFT} supplied as a network input and adjustable at inference from one model. It therefore shares single-pass compute and post-hoc control with CtrlFlow, and the two differ in where the control enters and in what the model costs to obtain. In GFT the strength enters the network, so each forward pass commits to one β_{GFT}, the range $\beta_{\text{GFT}} \in [0, 1]$ covers only $s \geq 0$, and the output is one opaque guided prediction. In CtrlFlow the scale multiplies $\mathbf{g}$ after the forward pass, so one evaluation yields the whole family $\{\mathbf{v}_0 + s\,\mathbf{g} : s \in \mathbb{R}\}$ together with $\|\mathbf{g}\|$ and $\cos(\mathbf{v}_0, \mathbf{g})$. GFT also fine-tunes the full backbone, 675M parameters for DiT-XL/2, on the training distribution itself, whereas CtrlFlow trains two rank-64 adapters (34M parameters) from teacher outputs and leaves the backbone unchanged.

We compare against GFT's released DiT-XL/2 ImageNet-256 checkpoint with our FID-50K evaluation at matched NFE. GFT runs its own sampler at N=25, and CtrlFlow uses DPM-Solver-2. Because β_{GFT} and s parameterize different families, each method is swept over its own control, and $\beta_{\text{GFT}} = 1/(1+s)$ serves only as a nominal correspondence.

Table 21: CtrlFlow against the released GFT checkpoint, FID-50K (↓) at N=25 and matched NFE.

Control setting	GFT (sweep β_{GFT})	CtrlFlow (sweep s)
Weak guidance, $s \approx 1$	**9.60**	9.87
$s \approx 2$	3.18	**3.07**
$s \approx 4$	8.10	**7.51**
$s \approx 8$	21.40	**19.28**

The two methods are close at weak guidance, where GFT is 0.27 FID better, and CtrlFlow leads as the scale grows, by 0.11 at $s \approx 2$, 0.59 at $s \approx 4$, and 2.12 at $s \approx 8$. We read this as comparable sample quality with a difference in what each checkpoint costs and exposes. GFT reaches this range by retraining all backbone parameters on ImageNet, while CtrlFlow reaches it with adapters distilled from teacher outputs on an unmodified backbone and additionally exposes $\mathbf{v}_0$ and $\mathbf{g}$ at every step.

D.17 Text-to-Image Pilot

The main evaluation is class-conditional, whereas post-hoc scale control is most often used in text-to-image generation. DiT-XL/2 already operates in the latent space of the Stable Diffusion autoencoder, so the mechanism is exercised under latent diffusion, and this pilot adds text conditioning. We apply the CtrlFlow recipe unchanged to Stable Diffusion 2.1 [Rombach et al., 2022], distilling CFG into two LoRA adapters following the adapter-distillation protocol of Jensen and Sadat [2025], and compare it with a scale-conditioned opaque student trained under the same protocol. Evaluation uses zero-shot COCO [Lin et al., 2015] FID-10K and CLIPScore [Hessel et al., 2021] at N=25 and s=7.5.

Table 22: Stable Diffusion 2.1 pilot. Zero-shot COCO FID-10K (↓) and CLIPScore (↑) at N=25, s=7.5.

Method	NFE	FID-10K	CLIPScore
CFG, two-pass	$2N$	9.82	0.309
Scale-conditioned opaque student	N	11.37	0.300
CtrlFlow	N	10.41	0.304

The ordering found on ImageNet holds under text conditioning. CtrlFlow recovers most of the gap to two-pass CFG at half the evaluations and stays ahead of the matched opaque student on both metrics. Because this is a single-configuration pilot rather than a grid study, we state the evaluated scope as class-conditional ImageNet-256 on two backbones plus this text-to-image pilot.

D.18 Decomposition Fidelity

Table 23 compares the learned components against teacher-generated ground truth via cosine similarity across four timesteps (1K samples, s=2).

Table 23: Decomposition fidelity. Cosine similarity (↑) between learned and teacher-generated components, averaged over 1K samples at s=2.

Component	t=0.8	t=0.5	t=0.2	t=0.05
$\cos(\hat{\mathbf{v}}_0, \mathbf{v}_0^*)$	0.942	**0.968**	0.957	0.911
$\cos(\hat{\mathbf{g}}, \mathbf{g}^*)$	0.836	**0.924**	0.908	0.867

Both components achieve high cosine similarity (> 0.83), with fidelity peaking at t=0.5 where guidance is most active. The $\mathbf{v}_0$ adapter consistently outperforms the $\mathbf{g}$ adapter by a margin of 0.04 to 0.11, a gap that arises because the unconditional velocity aligns closely with the frozen backbone's prior.

D.19 Decomposition Functional Equivalence

The cosine similarity in Table 23 measures geometric alignment between learned and teacher components. Section 4.4 reports the component-swap test, which asks whether the learned components can substitute for the teacher's without degrading generation quality. Because that test validates generation-level substitution, this appendix adds a complementary consistency check on the role of the base-flow branch.

Cross-Scale $\mathbf{v}_0$ Consistency. The base velocity $\mathbf{v}_0$ is the unconditional component and should not depend on the guidance scale s. Table 24 measures the cosine similarity between $\hat{\mathbf{v}}_0$ extracted at different guidance scales during inference. High similarity across scales (cosine > 0.97 for all scale pairs) supports the claim that the learned $\hat{\mathbf{v}}_0$ captures a scale-invariant unconditional flow rather than absorbing scale-dependent information. The small deviations reflect trajectory-level differences, since different s values produce different $\mathbf{x}_t$ sequences and the model therefore sees different inputs at each step.

Table 24: Cross-scale $\mathbf{v}_0$ consistency. Cosine similarity (↑) between $\hat{\mathbf{v}}_0$ extracted at different guidance scales, averaged over 1K samples at t=0.5.

	s=2 vs. s=4	s=2 vs. s=8	s=4 vs. s=8
$\cos(\hat{\mathbf{v}}_0^{(s_a)}, \hat{\mathbf{v}}_0^{(s_b)})$	0.987	0.971	0.983

D.20 Anchor Ablation: Unconditional versus Conditional Base

We thank an anonymous reviewer for suggesting this comparison. With $\mathbf{g} = \mathbf{v}_{\text{cond}} - \mathbf{v}_{\text{uncond}}$, our form anchors the CFG family at the unconditional field, $\mathbf{v}_0 + s\,\mathbf{g}$, while the alternative anchors it at the

conditional field, $\mathbf{v}_{\text{cond}} + (s-1)\,\mathbf{g}$, the original convention of Ho and Salimans [2021] with $w = s-1$. With exact fields the two give the same velocity at every s, and each recovers the other's base without an extra evaluation, since $\hat{\mathbf{v}}_{\text{cond}} = \hat{\mathbf{v}}_0 + \hat{\mathbf{g}}$ and $\hat{\mathbf{v}}_0 = \hat{\mathbf{v}}_{\text{cond}} - \hat{\mathbf{g}}$. Trained fixed-rank branches need not coincide, however, because adding or subtracting two learned outputs also combines their errors. A conditional anchor may also fit the conditioned backbone features more easily, and its unmodulated error $\boldsymbol{\epsilon}_{\text{cond}} + (s-1)\,\boldsymbol{\epsilon}_g$ carries a smaller guidance coefficient than ours. The deployed modulation, by contrast, uses the base flow directly, $\hat{\mathbf{v}}_0 + s\gamma\,\hat{\mathbf{g}}$ with $a = \cos(\hat{\mathbf{v}}_0, \hat{\mathbf{g}})$, whereas the exact analogue under the conditional anchor is $\hat{\mathbf{v}}_{\text{cond}} + (s\gamma - 1)\,\hat{\mathbf{g}}$ with $a = \cos(\hat{\mathbf{v}}_{\text{cond}} - \hat{\mathbf{g}},\, \hat{\mathbf{g}})$, so its gated base is obtained by subtraction.

We train the conditional-anchor student with the identical rank, curriculum, teacher, optimizer, and budget, using the composed loss $\|\hat{\mathbf{v}}_{\text{cond}} + (s-1)\,\hat{\mathbf{g}} - \mathbf{v}_{\text{cfg}}^{\text{teacher}}\|^2$. Table 25 first removes modulation and warping to isolate the parameterization, and Table 26 then applies the deployed path.

Table 25: Anchor ablation without modulation or warping (Core), FID-50K (↓) on shared class-balanced seed lists.

Anchor	$(25, 1)$	$(25, 2)$	$(8, 4)$	$(25, 8)$	$(4, 8)$
Unconditional, $\mathbf{v}_0 + s\,\mathbf{g}$ (ours)	9.92	3.14	11.48	23.65	73.41
Conditional, $\mathbf{v}_{\text{cond}} + (s-1)\,\mathbf{g}$	9.84	3.18	11.44	23.52	72.80

At the Core level the per-cell gaps of 0.04 to 0.61 FID are of the same order as the per-seed standard deviations of CtrlFlow-Full on these cells (0.07 to 0.51, Table 11), so we read the two anchors as matching within seed variation. The conditional anchor is marginally lower at s=1, where its first branch targets $\mathbf{v}_{\text{cond}}$ directly, and at high s, consistent with its smaller guidance coefficient. This parity is what the exact equivalence of the two forms predicts.

Table 26: Anchor ablation under the deployed modulation and warping path, FID-50K (↓).

Anchor	$(25, 8)$ Full	$(4, 8)$ Full
Unconditional (ours)	**19.28**	**48.95**
Conditional	20.41	51.58

Under the deployed path the anchors diverge by 1.1 to 2.6 FID in favor of the unconditional form, because the alignment gate receives the base flow directly in our form and by subtraction in the conditional one. Component diagnostics on fixed $(\mathbf{x}_t, c, t)$ states agree with this gap. At t=0.5 our learned $\hat{\mathbf{v}}_0$ reaches cosine 0.968 to the teacher's $\mathbf{v}_0$, while the conditional anchor's derived base $\hat{\mathbf{v}}_{\text{cond}} - \hat{\mathbf{g}}$ reaches 0.941. A condition-invariance probe also addresses whether the unconditional branch ignores useful conditioning. Fixing $(\mathbf{x}_t, t)$ and varying c, our $\hat{\mathbf{v}}_0$ keeps pairwise cosine similarity of at least 0.98 across conditions, so the branch learns a nearly condition-independent base flow rather than absorbing the conditioning signal. The conditional anchor is therefore a legitimate alternative whose Core quality matches ours within noise, and we keep the unconditional anchor because the deployed modulation uses the base flow directly.

D.21 Qualitative Comparison and Visualizations

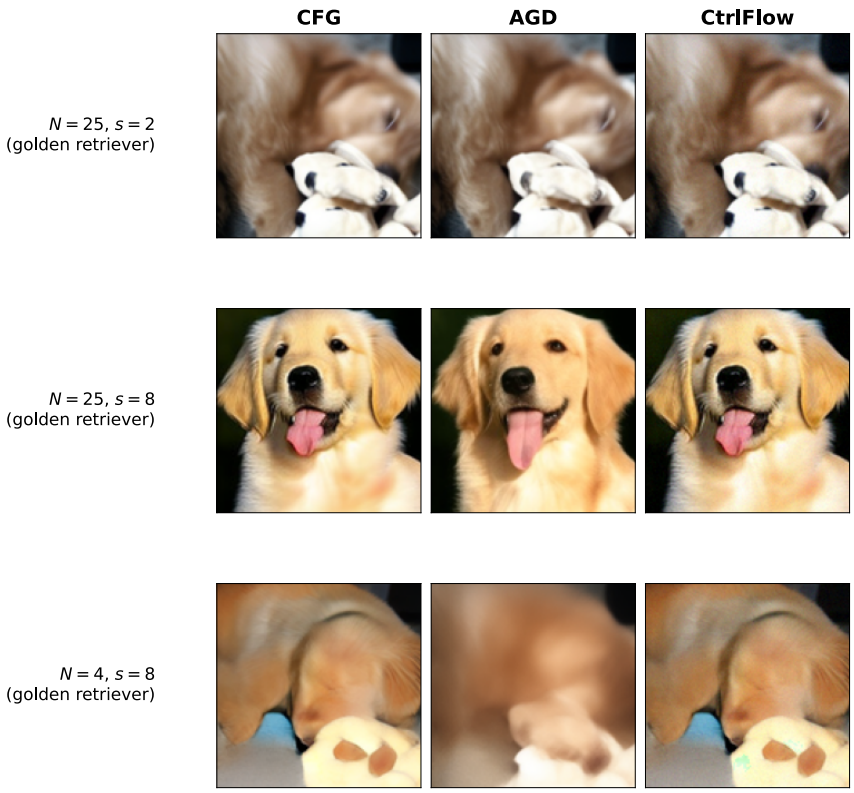

Figure 5: Qualitative comparison at three (N, s) operating points (same ImageNet class, *golden retriever*, per row). Rows show $(N{=}25, s{=}2)$, $(N{=}25, s{=}8)$, and $(N{=}4, s{=}8)$, and columns show CFG ($2N$ passes), the fixed-scale AGD variant ($s{=}2$, N passes, labeled "AGD"), and CtrlFlow (N passes). At $(25, 2)$ all methods produce comparable quality. At $(25, 8)$ AGD exhibits color saturation and loss of detail. At $(4, 8)$ AGD collapses to noisy outputs while CtrlFlow preserves recognizable structure. Aggregate FID values across all 1000 ImageNet classes are reported in Table 2, and additional classes are in Figure 6.

Figure 5 presents generated samples from CFG, the fixed-scale AGD variant, and CtrlFlow at three operating points. At $(25, 2)$, all methods produce visually similar outputs, consistent with the small FID gaps in Table 2. At $(25, 8)$, the fixed-scale variant's samples suffer from oversaturation and loss of fine detail, reflecting its Recall collapse to 0.14. CtrlFlow samples remain visually comparable to CFG at this scale. The contrast is most pronounced at $(4, 8)$, where the fixed-scale variant degenerates to noisy outputs (FID 82.17) while CtrlFlow preserves recognizable structure (FID 48.95).

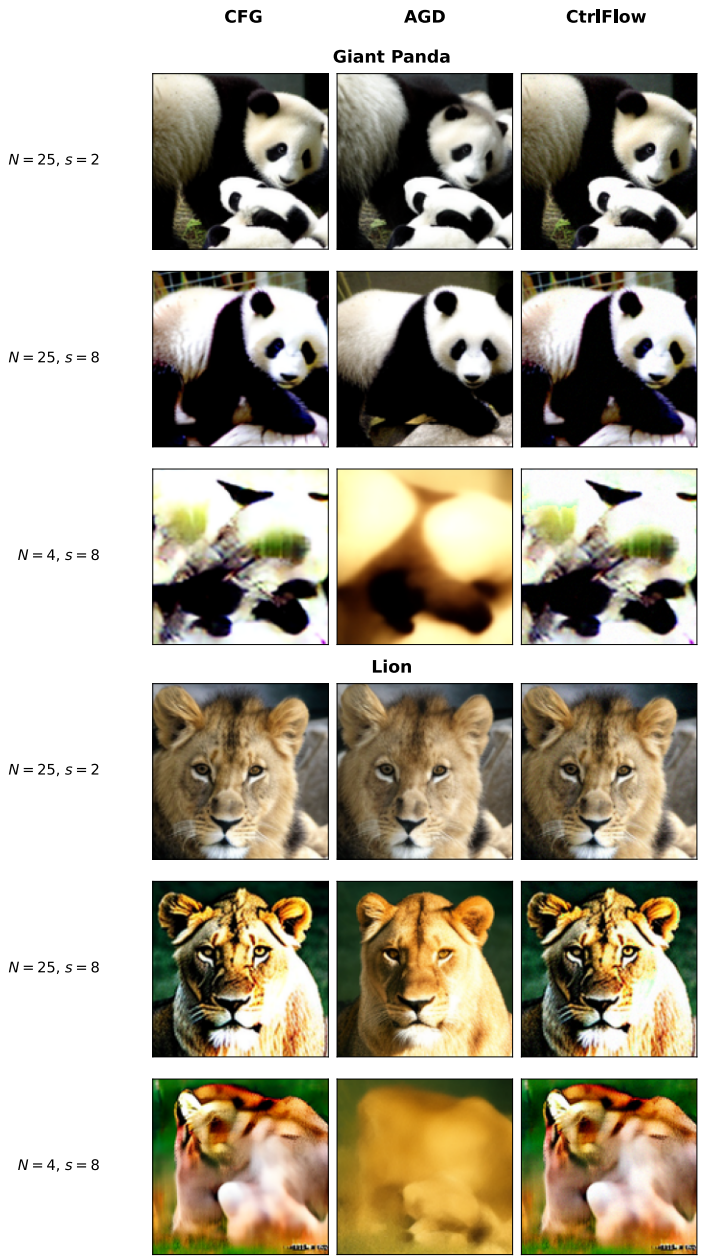

Figure 6: Extended qualitative comparison on two additional ImageNet classes (*giant panda*, *lion*) across the same three (N, s) operating points and three methods as Figure 5. Each block of three rows corresponds to one class, and columns show CFG ($2N$ passes), the fixed-scale AGD variant (N passes, labeled "AGD"), and CtrlFlow (N passes) at a shared seed. The visual story is consistent across classes. AGD oversaturates and collapses at large s, CtrlFlow tracks CFG closely, and the severity tracks the FID gaps reported in Table 2.

D.22 Extended Interpretability Maps

Figure 3 in the main text shows the guidance norm and alignment maps for *golden retriever*. Figure 7 extends this analysis to two additional classes, *butterfly* and *volcano*, to test whether the observed patterns generalize.

The temporal progression is consistent across both additional classes. At t=0.8, guidance norms are spatially diffuse regardless of class identity. The model has not yet resolved object structure

at this stage, so conditioning acts as a global bias. By t=0.2 to t=0.05, guidance concentrates on class-discriminative regions. For *butterfly*, these are wing patterns and body contours. For *volcano*, they are the crater rim and smoke plume. Together with the golden retriever maps in the main text, where guidance focuses on facial features and fur texture, the class-specific spatial concentration confirms that the learned $\mathbf{g}$ encodes semantically meaningful conditioning adjustments rather than a generic sharpening signal.

The alignment maps reveal the same orthogonal-to-aligned progression in both classes. At early timesteps, $\cos(\mathbf{v}_0, \mathbf{g}) \approx 0$, and as the trajectory converges, alignment increases toward 1. The consistency of this pattern across three visually distinct categories (fine-grained texture, object with background, scene-level structure) supports the claim that the decomposition captures a general structural property of the guided velocity field.

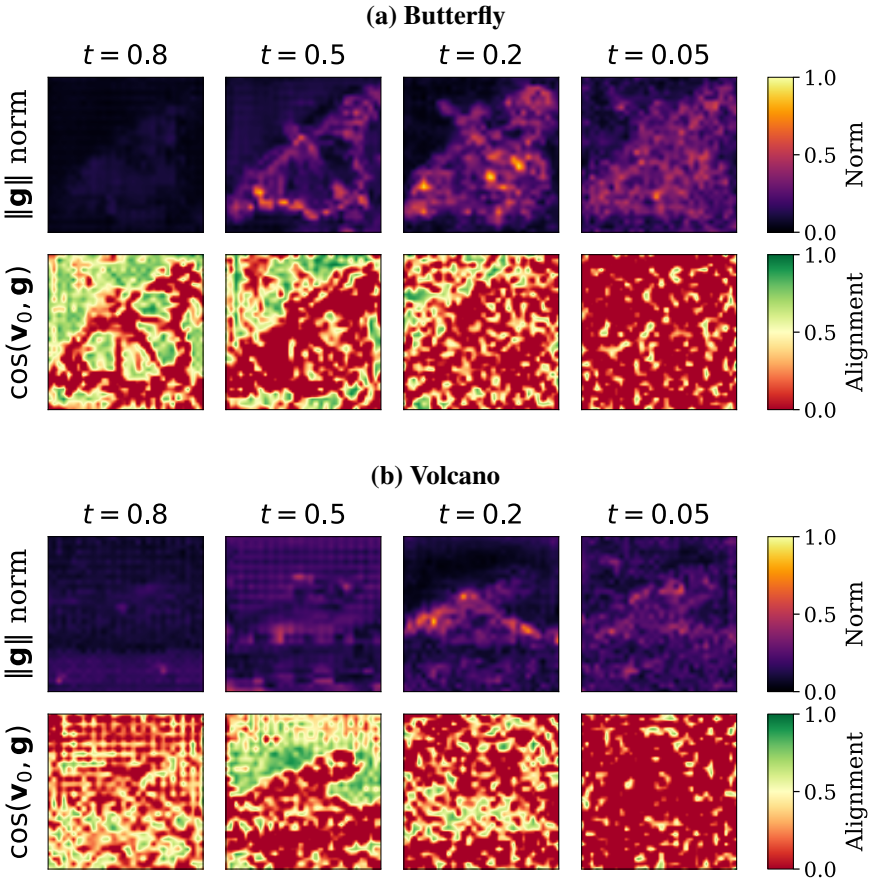

Figure 7: Cross-class interpretability maps (extending Figure 3, which shows *golden retriever*). Each panel shows guidance norm $\|\mathbf{g}\|$ (top row) and alignment $\cos(\mathbf{v}_0, \mathbf{g})$ (bottom row) at four timesteps. The temporal pattern matches the main-text figure. Guidance is diffuse at t=0.8 and concentrates on class-discriminative regions by t=0.05. The spatial focus varies by class, with wing patterns for *butterfly* and crater and smoke for *volcano*, confirming that $\mathbf{g}$ encodes class-specific semantic structure.

D.23 Quantitative Localization of Guidance Maps

The maps above are qualitative and cover few classes. To test whether guidance concentrates on the conditioned object, we compute guidance-norm maps on forward-noised ImageNet validation images and score them against the ground-truth object boxes. We report pointing accuracy [Zhang et al., 2018], the fraction of images whose map maximum falls inside a box, energy-in-box, the fraction of map mass inside the boxes, and deletion and insertion AUC [Petsiuk et al., 2018], which track how classifier confidence changes as the highest-scoring pixels are removed from or added to the image.

All four scores are oriented so that higher is better. Grad-CAM [Selvaraju et al., 2017] maps and a uniformly random map serve as references.

Table 27: Localization of guidance-norm maps against ground-truth boxes (↑ for all columns).

Map	Pointing acc.	Energy-in-box	Deletion AUC	Insertion AUC
Guidance norm, CtrlFlow	**0.81**	**0.613**	**0.624**	**0.681**
Guidance norm, two-pass teacher	0.80	0.601	0.612	0.669
Grad-CAM	0.74	0.548	0.587	0.642
Random	0.14	0.219	0.501	0.503

The learned maps localize the object better than Grad-CAM on every score and closely track the maps computed from the two-pass teacher, which grounds the interpretability claim of Section 4.4 quantitatively.

D.24 Modulation and Warping Visualization

Figure 8 illustrates the three guidance modulation functions. The temporal window $w(t)$ narrows as s increases, concentrating guidance in the mid-timestep range. The dampening function $\alpha(s)$ engages only beyond $s_{\text{thresh}} = 4$, leaving low-scale generation unmodified. The alignment gate $\varphi(a)$ attenuates guidance when $\mathbf{v}_0$ and $\mathbf{g}$ are misaligned.

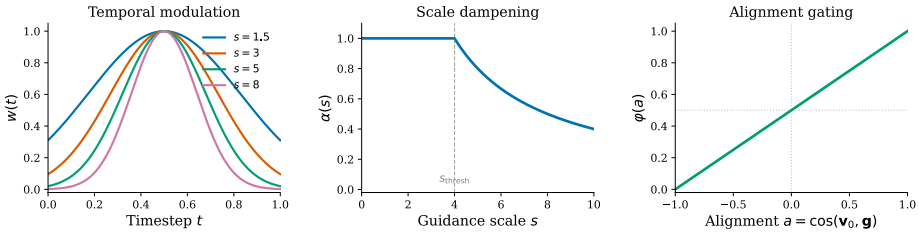

Figure 8: Guidance modulation profiles. **Left.** Temporal modulation $w(t)$ narrows with increasing s. **Center.** Scale dampening $\alpha(s)$ activates above s_{thresh}. **Right.** Alignment gating $\varphi(a)$ attenuates guidance under misalignment.

D.25 Time-Warp Visualization

Figure 9 visualizes the time-grid warping curves at different guidance scales. At s=1, the schedule remains uniform. At s=8, the dilation weight reaches $1.5\times$ at the trajectory endpoints, concentrating solver evaluations in the mid-timestep region where the guided velocity changes most rapidly.

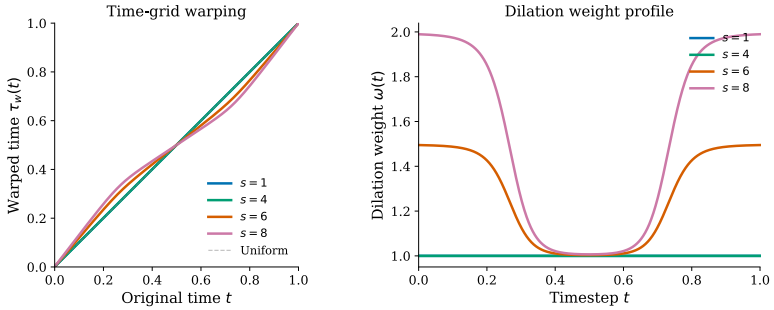

Figure 9: Time-grid warping. **Left.** Warped time $\tau(t)$ deviates from the uniform diagonal at high s. **Right.** Dilation weight $\omega(t)$ increases at trajectory endpoints for large guidance scales.

D.26 FID vs. NFE

Figure 10 plots FID against NFE at s=4, confirming that CtrlFlow at NFE $= N$ matches CFG at NFE $= 2N$.

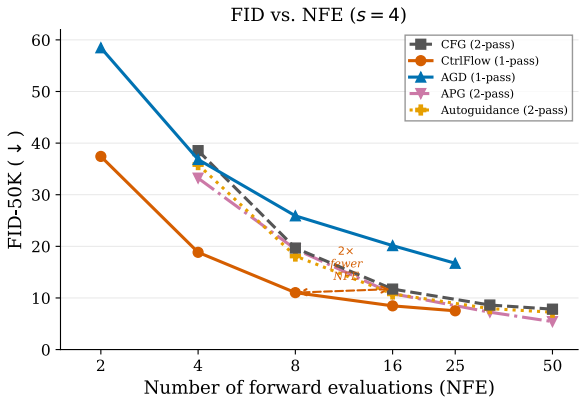

Figure 10: FID-50K vs. NFE at s=4. CtrlFlow (solid orange) at NFE $= N$ achieves comparable FID to CFG (dashed gray) at NFE $= 2N$, halving compute for equivalent quality.

E Reproducibility

This appendix specifies the baseline protocol, computational cost breakdown, and implementation details required to reproduce every result in Section 4.

E.1 Baseline Protocol

Table 28 records the code source, training configuration, and evaluation scope for each baseline. Every two-pass baseline runs its official public implementation with default hyperparameters on DiT-XL/2 ImageNet 256×256. Both AGD variants are trained by us, as described in the protocol note below, and GFT is evaluated from its released checkpoint. We tuned CtrlFlow's modulation and warping hyperparameters on a 5K-image validation set described in Appendix C.

Table 28: Baseline protocol. All methods use DiT-XL/2 on ImageNet 256×256. "Train s" is the guidance scale used during training. "Eval s" is the range tested. "s-free" indicates whether one checkpoint supports variable s at inference without retraining.

Method	Source	Hyperparams	Train s	Eval s	Models	s-free
CFG	Official DiT	Default	N/A	$[1, 12]$	1	✓
CFG-Zero*	Official repo	Default	N/A	$[1, 8]$	1	✓
APG	Official repo	Default	N/A	$[1, 8]$	1	✓
Autoguidance	Official repo	Default	N/A	$[1, 8]$	1	✓
CFG + Interval	Official repo	Default	N/A	$[1, 8]$	1	✓
AGD (fixed s)	Ours, AGD design	Ablation	2.0	$[1, 8]$	1	×
AGD-MultiScale	Ours, AGD design	Matched	$\mathcal{U}[1, 8]$	$[1, 8]$	1	✓
GFT	Released ckpt.	Default	$\beta_{\text{GFT}} \in [0, 1]$	$[1, 8]$	1	✓
CtrlFlow	Ours	Tuned (App. C)	$\mathcal{U}[1, 8]$	$[1, 12]$	1	✓

AGD Protocol Note. As published, AGD [Jensen and Sadat, 2025] is scale-conditioned. Its adapter receives the guidance scale as an embedded network input and is trained over a sampled range of scales ($\mathcal{U}[1, 6]$ for DiT-XL/2), so one checkpoint serves many scales. No DiT-XL/2 AGD checkpoint is publicly available, so we train two variants on our frozen teacher and trajectories. AGD-MultiScale follows the published design, with the scale range widened to $\mathcal{U}[1, 8]$ to match our training range, rank 128 (34M parameters) to match CtrlFlow's capacity, and the same curriculum and optimizer. It is the faithful AGD baseline and the primary single-pass comparison. AGD (fixed s=2) is an ablation of that design trained at a single scale (14M parameters). We evaluate it at other scales to measure single-checkpoint behavior outside the training scale, which is not how AGD is intended to be used,

and its high-scale failure should be read in that light. The submitted version of this paper described this row as AGD's public s=2 checkpoint. It is our own single-scale training of the AGD design, and this version corrects the description.

E.2 Computational Cost

Table 29: Computational cost comparison on DiT-XL/2 at N=25.

Method	Params	Extra	NFE	ms/img	Train (GPU-hr)
CFG	675M	0%	$2N$	843	0
CFG-Zero*	675M	0%	$2N$	854	0
APG	675M	0%	$2N$	852	0
Autoguidance	1015M	50%	$2N$	891	120
AGD (fixed s)	689M	2%	N	432	32
CtrlFlow	**709M**	**5%**	$\boldsymbol{N}$	**437**	**48**

Latency Breakdown. The two branches share one frozen backbone pass per step, so backbone compute is incurred once and NFE $= N$ is exact. The speedup is $1.93\times$ rather than $2\times$ because half of CFG's 843 ms, two sequential evaluations per step, is 421.5 ms, and the adapters and modulation add 15.5 ms at N=25 (Table 30). The single-adapter AGD baseline (Table 29) measures 432 ms under the same protocol, so the second branch and the gates account for the remaining 5 ms.

Table 30: Latency breakdown of CtrlFlow at N=25, batch size 1, A100, 50 warmup and 200 timed iterations under CUDA synchronization.

Component	ms/image
Frozen backbone, 25 steps at 16.9 ms	421.5
Adapter matrix multiplications, both branches	12.6
Modulation and composition	2.9
Warp precomputation (cached)	< 1
Measured total	**437**

E.3 Implementation Details

LoRA Placement. Both adapters attach rank-64 LoRA modules to the query, key, value, and output projections of every self-attention layer in the DiT backbone. MLP layers and layer normalization parameters remain frozen. The two branches do not keep separate hidden states. The frozen backbone computes one hidden-state stream per step, each LoRA module reads that shared stream, and each branch's low-rank residuals are accumulated into its own velocity-shaped output rather than written back into the stream. The two outputs are combined as $\hat{\mathbf{v}}_0 + s \cdot \hat{\mathbf{g}}$ after the forward pass.

Teacher Target Construction. For each training sample $(\mathbf{x}_t, c, t, s)$, the frozen teacher performs two forward passes, one unconditional ($c = \varnothing$) and one conditional (c). The resulting teacher target is $\mathbf{v}_{\text{cfg}}^{\text{teacher}} = \mathbf{v}_\theta(\mathbf{x}_t, \varnothing, t) + s \cdot [\mathbf{v}_\theta(\mathbf{x}_t, c, t) - \mathbf{v}_\theta(\mathbf{x}_t, \varnothing, t)]$. Teacher outputs are generated online rather than cached.

Solver and Evaluation. Inference relies on DPM-Solver-2 (second-order, multistep) with velocity prediction and the uniform-then-warped timestep schedule. FID-50K is computed through the ADM evaluation suite using the standard ImageNet-256 Inception statistics file. All methods share the same three 50K class-balanced generation seed lists.

Precision and Hardware. Training and inference run in bf16 mixed precision with TF32 enabled on 8×A100 (80 GB). The AGD-MultiScale baseline and 3-seed evaluation runs were executed on 4×H200 (141 GB) via Modal cloud compute. All methods share the same ADM evaluation pipeline, class-balanced seed lists, and precision settings. Small hardware-dependent numerical differences

may remain across GPU architectures. Timing measurements use the A100 setup with batch size 1, 50 warmup iterations, and 200 timed iterations under CUDA synchronization.

Validation Protocol. Modulation and warping hyperparameters were tuned on a held-out 5K-image validation set spanning five (N, s) configurations, namely $(25, 2)$, $(25, 8)$, $(8, 4)$, $(4, 2)$, and $(4, 8)$. These five configurations overlap with the headline test configurations. We acknowledge this overlap as a limitation. Two checks use configurations never seen during selection. CtrlFlow matches CFG on all 20 untuned grid configurations except $(2, 8)$, and the mechanism-transfer test of Appendix D.12 includes the untuned $(16, 6)$ and $(8, 6)$. The ablation tables in Appendices D.8 to D.11 provide further robustness evidence across hyperparameter variations.

Additional Evaluations. Class adherence uses a frozen torchvision ResNet-50 (IMAGENET1K_V2) on the FID seed lists (Appendix D.5). The GFT comparison uses the released DiT-XL/2 ImageNet-256 checkpoint with its own sampler at matched NFE (Appendix D.16). The text-to-image pilot uses Stable Diffusion 2.1 with zero-shot COCO FID-10K and CLIPScore (Appendix D.17).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Main claims in the abstract and introduction (96% CFG-match accuracy against 20 of 25 for a matched scale-conditioned opaque student, $1.93\times$ speedup, interpretable decomposition) are supported by experiments in Section 4, ablations in Section 4.3, and the matched-baseline, mechanism-transfer, class-adherence, and text-to-image results in Appendix D.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 includes an explicit Limitations paragraph discussing two-backbone evaluation (DiT-XL/2 and DiT-L/4), text-to-image evidence limited to a single-configuration Stable Diffusion 2.1 pilot, untested video and higher resolutions, the overlap between tuning and headline configurations, and the deliberate departure from CFG dynamics at high guidance scales.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Appendices B.1, B.2, and B.3 provide supporting analysis with stated assumptions. Formal results include gradient scaling under the unified loss (Claims 1 to 3), decomposition identifiability under scale variation (Proposition 1), per-step velocity error bounds (Propositions 2 and 3), the exact bias-error decomposition of the modulated velocity (Proposition 4), schedule monotonicity (Proposition 5), and curvature-weighted step allocation (Proposition 7). All assumptions are stated explicitly. The identifiability result assumes realizability of the adapter function classes. In practice, low-rank adapters provide an approximation, and decomposition fidelity is validated empirically (Section 4.4). The Jacobian-comparability condition behind the gradient-scaling claim is measured empirically (Appendix B.1).

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4.1 specifies the backbone, dataset, evaluation protocol, and training details. Appendix C provides all hyperparameter values, and Appendix E specifies the baseline protocol, cost measurement, and implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code is not included with the paper. ImageNet is publicly available, and the paper and appendix provide sufficient detail for independent reimplementation.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4.1 specifies the LoRA rank, curriculum, and training budget, and Appendix C provides the optimizer, learning rates, batch size, phase durations, and the complete hyperparameter table.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All 25 configurations are evaluated over 3 generation seeds. Standard deviations are reported in Appendix D.6 (Table 11). Representative stds range from ± 0.07 at easy configs to ± 0.94 at the extreme $(2, 8)$ failure case. The 24/25 CFG-match classification is stable across all seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Training requires 48 GPU-hours on $8\times$A100 (80 GB). Inference timing is reported in Table 29. Total project compute, including preliminary experiments, baselines, evaluation, and the experiments added during review, was approximately 1,400 GPU-hours, part of which ran on Modal cloud compute supported by Modal academic research credits.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: This work conforms to the NeurIPS Code of Ethics. It proposes an efficiency improvement for existing diffusion models and does not introduce new risks beyond those inherent to image generation.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: See the Broader Impacts paragraph in Section 5. The method reduces compute cost for guided diffusion but inherits the same misuse risks as any image generation technology.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: This paper proposes an inference-efficiency method and does not release a pretrained model or dataset. Safeguards are the responsibility of the base model provider.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: DiT-XL/2 is released under a permissive license. ImageNet is used under its standard research license. Stable Diffusion 2.1 is used under the CreativeML Open RAIL++-M license, COCO under its published terms of use, and the torchvision ResNet-50 weights and the released GFT checkpoint under their respective licenses. All baseline implementations and checkpoints are cited with their original papers.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets or pretrained models are released with this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as a component of the core method. Any use of LLMs for writing assistance does not impact the methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.