FlashControl: One-Step Controllable Generator via Distillation-Friendly Single-Stream Teachers

Ngan Nguyen[1*], **Dung Nguyen**[1*], **Quan Dao**[2],
Dimitris N. Metaxas[2], **Minh Hoai**[1,4], **Cuong Pham**[1,3], **Anh Tran**[1]

[1]Qualcomm AI Research [2]Rutgers University
[3]Posts and Telecommunications Institute of Technology [4]Adelaide University

{nganguy,dungtien,minhhoai,pcuong,anhtra}@qti.qualcomm.com
quan.dao@rutgers.edu, dnm@cs.rutgers.edu



Figure 1: Our controllable image generator, **FlashControl**, turns spatial conditions into high-quality images in just **a single step.**

Abstract

Diffusion models have become a leading approach in visual generative modeling, particularly effective in text-to-image and controllable generation. Their main drawback is slow inference as it requires many sampling steps. While recent distillation methods enable one-step text-to-image generation, extending these techniques to controllable generation is still challenging. Existing controllable models, such as ControlNet, rely on dual-branch design that make distillation difficult. In this work, we adopt a simpler strategy. Instead of adding extra conditional branches, we finetune a selected subset of layers in the original text-to-image diffusion model to directly enable controllable generation. This results in **DeltaControl**, a single-flow, multi-step diffusion model capable of supporting spatial-condition generation while matching the performance of dual-branch approaches. Building on this teacher model, we further apply step-distillation to obtain **FlashControl**, a one-step model capable of spatial controllable generation with a single forward pass.

1 Introduction

Diffusion models [33, 5, 15, 14, 40] have become a strong paradigm for both text-to-image generation and controllable synthesis. In parallel, recent distillation methods [8, 48, 49, 26, 37, 36, 29] have

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

reduced diffusion sampling to a small number of steps, and one-step generators are now increasingly competitive with their multi-step teachers. Extending this progress to controllable generation, however, is substantially less direct. Existing controllable models such as ControlNet and T2I-Adapter inject spatial guidance through external control pathways on top of a pretrained text-to-image backbone [51, 27]. These designs are effective in the standard multi-step setting, but our analysis suggests that they are poorly matched to one-step distillation. When such a teacher is compressed into a single forward-pass student, the student must absorb both the base generative process and the control-specific correction into one computation. In practice, this leads to weak control, unstable optimization, or blurry results, even when the multi-step teacher itself performs well.

This observation motivates the first hypothesis of our work: *for one-step controllable generation to succeed, control must be integrated into the teacher backbone in a form that remains transferable under distillation, rather than attached through an auxiliary branch*. Based on this view, we introduce **DeltaControl**, a single-stream controllable teacher designed explicitly for one-step transfer. Instead of duplicating large portions of the network, DeltaControl adapts a selected subset of parameters in the original backbone to encode spatial control. We study this design on both UNet and DiT backbones and find that tuning feed-forward layers provides the most favorable behavior for controllability and convergence under a matched parameter budget. Our interpretation is straightforward: in a single-stream architecture, attention layers largely govern global feature mixing, so perturbing them can induce broad and unstable changes, whereas feed-forward layers act on already mixed representations and provide a more stable subspace for injecting control. This yields a controllable teacher that is compact, single-stream, and compatible with standard distillation. More importantly, it distinguishes our goal from prior single-stream controllers such as RepControlNet [9] and OminiControl [44]: our objective is not only efficient controllable generation, but a *distillation-friendly teacher architecture* whose control mechanism survives compression into a one-step student.

Building such a teacher is only the first step. The next challenge is to distill it into a one-step generator without obscuring the effect of teacher architecture. Recent methods such as SwiftBrush [29, 8] and DMD [48, 49] show that score-based distribution matching can transfer a multi-step diffusion model to a one-step generator efficiently [49, 8]. In this work, we adopt **vanilla DMD** [49] as the main distillation objective because it is a simple and widely used score-based formulation that cleanly isolates the role of the teacher architecture. We leave stronger adversarial variants, such as DMD2 [48], to the appendix. This choice is deliberate: our main claim is that teacher design is the critical ingredient for one-step controllable distillation, and we therefore avoid introducing extra adversarial components in the main setting that could confound this comparison.

Even with a suitable teacher and a simple distillation objective, one-step controllable students are still fragile. Different controls, such as canny edges, depth maps, segmentation masks, and poses, have very different structures and scales. If we inject them naively, they can disturb the backbone features and make training unstable. We address this with **AutoAdapter**. AutoAdapter first aligns condition features to the hidden-state statistics, then applies token-wise gating to control how strongly each token is injected. In our framework, AutoAdapter is not just a small add-on. It is the module that makes different control types usable within one model and keeps early training stable, when the student is most sensitive to feature mismatch.

A separate issue appears during one-step distillation itself. During late-stage distillation, the student starts ignoring variations in its input Gaussian noise, causing different noise samples under the same prompt-condition pair to produce nearly identical outputs. This hurts both diversity and fine detail. To address this, we introduce a **Latent Variance Preservation** objective, which encourages the output to retain a meaningful response to perturbations in the input noise. The role of this term is specific to one-step diffusion distillation: it preserves stochastic sensitivity in the student and reduces the tendency toward deterministic collapse. Combined with pure DMD and AutoAdapter, it gives a training recipe that is stable enough for one-step controllable generation.

Together, these components form a unified view of the problem. DeltaControl addresses the architectural mismatch between controllable teachers and one-step students. Single stream teacher with pure DMD loss keeps the distillation objective tied to the spatial condition. AutoAdapter aligns heterogeneous controls to the backbone feature space. Latent Variance Preservation keeps the one-step student responsive to noise. Under this formulation, FlashControl is not a loose collection of existing modules. It is a pipeline in which each component resolves a specific failure mode that appears when controllable generation is compressed into a single forward pass.

Our contributions are as follows:

- We show that naive one-step distillation fails for dual-branch controllable models, while single-flow teachers provide a better foundation for one-step controllable generation.
- We introduce **DeltaControl**, a single-stream controllable teacher that integrates structural control into the base backbone by adapting a selected subset of layers. Across both UNet and DiT backbones, we find that feed-forward tuning provides a favorable balance between controllability, efficiency, and distillation compatibility.
- We show that a DMD-based distillation objective, together with **AutoAdapter**-based condition alignment, provides a stable recipe for one-step controllable generation and preserves spatial faithfulness under structural controls.
- We introduce a **Latent Variance Preservation** objective for one-step controllable distillation, which reduces deterministic collapse by maintaining the student's sensitivity to input noise, and we demonstrate real-time one-step controllable synthesis across multiple control modalities within a single model.
- To the best of our knowledge, **FlashControl** is among the first one-step controllable distillation frameworks to explicitly build the controllable teacher as a LoRA-compatible single-stream backbone adaptation, rather than distilling from a duplicated control branch.

2 Related Work

Diffusion Models [14, 34] have surpassed GAN [11] across a wide range of tasks, including controllable generation [51, 27], image editing [1, 25], and personalized synthesis [35]. This progress began with LDM [34] and was further advanced by the DiT architecture [30]. The recent text-to-image models, SD3 [10] and FLUX [15, 16], achieve impressive results by adopting Rectified Flow training [20], and Multi-Modal Transformer (MMDiT) backbone [30].

Multi-step Controllable Diffusion Models. Controllable synthesis guides text-to-image diffusion with structural inputs (e.g., edges, depth, segmentation, pose). T2I-Adapter [27] trains lightweight adapters on a frozen backbone, while ControlNet [51] adds a parallel control branch and has inspired many variants, including ControlNet-XS [50], UniControlNet [52], UniControl [31], and AnyControl [43]. To reduce tuning cost, CTRLORA [47] adapts small parameter subsets of the control branch. Instead of using dual-branch, some recent works, such as UniCombine [45] and OminiControl [44], append the conditional inputs to the noisy latent input and denoise them with a LoRA-finetuned single-branch DiT. These techniques often leverage strong backbones like FLUX and incur a massive computation cost growth with respect to the number of conditions. Despite strong controllability, these methods still rely on multi-step sampling, leading to slow inference.

Diffusion Distillation. Diffusion models are slow at inference due to many denoising steps, motivating distillation to shorten sampling while preserving quality [36, 21]. Consistency Models and LCM enable few-step sampling but still show a quality gap at one step [41, 22]. ADD/LADD use adversarial losses for 1-4 step distillation but can be unstable [37, 38], while DMD/DMD2 distill from the teacher score field for one-/few-step generation [49, 48]. SwiftBrush reduces overhead via image-free score distillation with VSD [28, 46], and SBv2 improves further with stronger initialization, efficient LoRA, and a clamped-margin CLIP loss, sometimes surpassing the teacher [8].

3 Preliminaries

Diffusion and Flow-based Generative Models. Modern text-to-image models generate images by learning to map a noisy sample back to clean data. Let x_0 denote a clean image or latent, $t \in [0, 1]$ a timestep, and $\epsilon \sim \mathcal{N}(0, I)$. A common noising process is

$$x_t = \alpha_t x_0 + \sigma_t \epsilon,$$

where α_t and σ_t define the signal and noise scales. Depending on the model family, the denoiser predicts either the noise ϵ or a velocity/flow target v_t. We write this generically as

$$f_\phi : (x_t, t, y) \mapsto \hat{r}_t, \qquad \mathcal{L}_{\text{gen}} = \mathbb{E}_{x_0, y, t, \epsilon} \left[\| f_\phi(x_t, t, y) - r_t \|_2^2 \right],$$

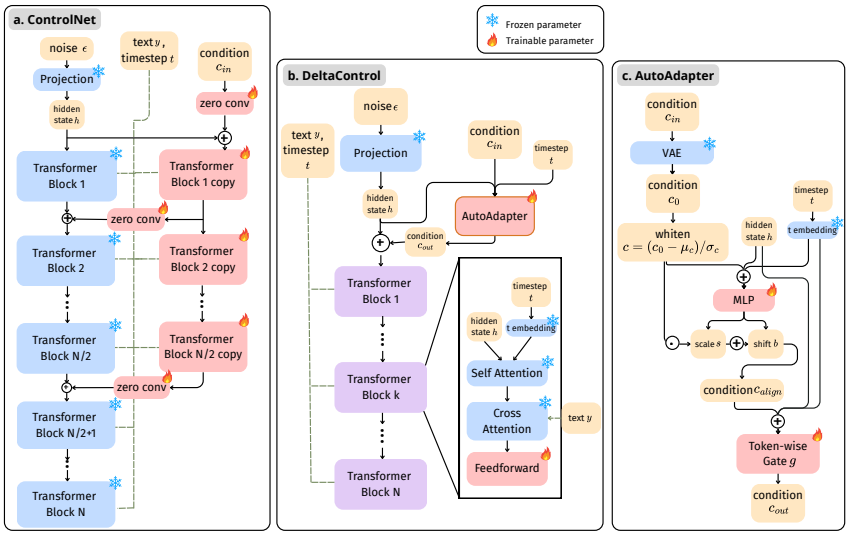


Figure 2: Comparison of ControlNet (a) and our proposal **DeltaControl** (b). The architecture of AutoAdapter is illustrated in (c).

where $r_t = \epsilon$ for noise-prediction diffusion models and $r_t = v_t$ for velocity or flow-based models. At inference, sampling starts from Gaussian noise and repeatedly applies the learned denoising or flow field, which yields high quality but requires many network evaluations.

ControlNet. Text prompts provide global semantic guidance, but many applications require spatial control from a condition map c, such as edges, depth, segmentation, or pose. ControlNet [51] addresses this by attaching an auxiliary control branch to a pretrained text-to-image backbone. The control branch processes (x_t, t, c) and injects condition-dependent residuals into the frozen base denoiser, usually through zero-initialized layers. This design preserves the pretrained generative capability while learning spatial alignment.

Distribution Matching Distillation. We use Distribution Matching Distillation (DMD) [49, 48] as our main one-step distillation objective. DMD trains a generator G_θ to match the distribution of a pretrained diffusion teacher without simulating the full teacher trajectory at inference. Given input noise z, text prompt y, and spatial condition c, the student predicts

$$\hat{x}_0 = G_\theta(z, y, c).$$

The student output is then noised to timestep t to obtain x_t. DMD updates the student using the score discrepancy between a frozen teacher score estimator ϵ_ϕ and a trainable fake score estimator ϵ_ψ:

$$\nabla_\theta \mathcal{L}_{\text{DMD}} = \mathbb{E}_{z,\epsilon,t,y,c}\left[w(t)\left(\epsilon_\phi(x_t, t, y, c) - \epsilon_\psi(x_t, t, y, c)\right) \frac{\partial G_\theta(z, y, c)}{\partial \theta} \right],$$

where $w(t)$ is a timestep-dependent weighting function.

4 FlashControl

4.1 Overview

Our objective is **one-step (single forward pass) controllable text-to-image generation via distillation**, with **stable optimization** and **high-quality spatial condition model**. We hypothesize that **one-step controllable generation is substantially harder than one-step text-only generation**, because the student must preserve both visual fidelity and spatial faithfulness within a single network evaluation. In particular, the common dual-branch design used by ControlNet-style methods is not well matched to one-step distillation, as discussed later in Section 4.2. To address this problem, we propose **FlashControl**, a two-stage teacher→student pipeline for real-time, high-quality and controllable generation.

Stage 1: distillation-friendly conditional teacher (DeltaControl in Section 4.3). One-step text-to-image distillation methods typically rely on a single-branch teacher architecture. Motivated by this observation, we train a single-branch DeltaControl teacher, which enables spatial signal control by fine-tuning all feedforward layers from the base text-to-image diffusion model. To better handle heterogeneous control modalities, we introduce **AutoAdapter**, a lightweight conditioning module.

Stage 2: one-step student (FlashControl) via distillation in Section 4.4. We distill the single-stream DeltaControl teacher into a one-step generator using **Distribution Matching Distillation (DMD)** [49, 48]. In addition, we deliberately **drop adversarial distillation**, since controllable generation requires realism under a spatial constraint rather than unconstrained image realism. Finally, to prevent late-stage collapse in one-step distillation, we introduce **Latent Variance Preservation (LSP)**, which maintain the output variation and prevents deterministic mapping.

4.2 From Dual-branch to efficient single-stream teacher.

Is dual-branch ControlNet optimal for one-step distillation? ControlNet [51] is the most common framework for multi-step controllable image generation. A natural approach for *one-step* controllable T2I is therefore to (i) train a ControlNet teacher and (ii) apply a one-step distillation method (e.g., DMD [48, 49]). We experiment with distilling dual-branch models, including ControlNet and T2I-Adapter, into onestep generators using DMD. However, the results show that this straightforward pipeline results in low-quality and blurry results, as in Fig. 4 and Table 5, even when the multi-step teacher itself performs well.

We attribute this failure to an architectural mismatch between *dual-branch teacher* and *single-step student*. ControlNet decomposes controllable generation into two coupled components: a base denoiser and an auxiliary control branch that injects condition-dependent residuals. In one-step distillation, the student must simultaneously approximate the base denoising dynamics *and* the control-induced correction within a single forward pass. This entangled objective leads to unstable optimization: the student either collapses to the base model (weak control) or overfits noisy control residuals (degrading fidelity). This behavior is consistent with recent observations in distilling large dual-branch architectures, where additional stabilization tricks are required and training may still collapse (e.g., gradient explosion) [7]. We further verify in Fig. 4 that simply "simplifying" the objective by freezing either the base denoiser or the control branch does not resolve the issue. Overall, these results suggest that ControlNet's dual-branch architecture is not well suited as a teacher for one-step controllable distillation.

Why DeltaControl? From RepControlNet to a unified single-branch teacher. This analysis suggests a simple principle: a strong one-step controllable student should be distilled from a single-branch controllable teacher, where control is integrated into the backbone rather than produced by an external branch. RepControlNet [9] is a representative approach in this direction. It reparameterizes control by replicating parts of the backbone and learning condition-dependent residual transformations that can be merged into a single network, removing the need for a dual control branch during inference.

We identify several limitations of RepControlNet. First, it replicates parts of the backbone during training following on a tedious replicate-and-merge strategy, leading to unnecessary increase in memory and computation cost. Second, it does not investigate which layers should be adapted to effectively incorporate control signals. Instead, it adopts a dense "replicate-and-tune" strategy that duplicates and finetunes many layers without analyzing their roles in controllability. Third, RepControlNet is evaluated only on the UNet backbone, leaving open the question of whether the approach generalizes to transformer-based architectures such as DiT. Finally, RepControlNet focuses primarily on a single-condition setting and evaluates only a limited set of control modalities. This restricts its applicability in scenarios where multiple heterogeneous control signals must be handled within a unified framework.

In contrast, our method addresses all four limitations. We do not replicate network components but directly finetune some of them, offer more efficient training and memory usage. We perform an analysis to identify which layers are most effective for adapting control signals, design a strategy that generalizes across both UNet and transformer backbones, and develop a unified controllable framework capable of handling diverse control modalities within a single model.

Comparison to other single-branch approaches. Recent methods, such as OminiControl [44] and UniCombine [45], append the conditions to the noisy latent input, offering a cleaner DiT-based single-stream formulation for versatile multi-step control. However, their goal is efficient controllable generation itself, not the construction of a teacher whose control interface can be faithfully compressed into a one-step student. In contrast, DeltaControl is designed around this stricter requirement: it uses **selective backbone adaptation** to integrate control directly into layers that remain stable under transfer, and combines this with AutoAdapter so that the student inherits a normalized, reliability-aware control interface during distillation. As a result, our single-stream design is not merely lighter; it is more friendly to distill, as will be showed in Section 5.

4.3 Stage 1: DeltaControl - Multi-step Conditional Model

4.3.1 Efficient Finetuning Strategy.

Inspired by RepControlNet, DeltaControl removes the dual-branch architecture and instead fine-tunes only a selected subset of parameters in the base T2I model. To determine which layers are most suitable for learning control signals, we conduct a series of experiments on two widely used diffusion backbones: UNet and DiT. For DiT, which consists of Transformer blocks combining self-attention, cross-attention, and feed-forward (FFN) layers, we evaluate three finetuning configurations with the same number of trainable parameters as vanilla ControlNet: (i) attention-only, (ii) FFN-only, and (iii) a mixed subset of attention and FFN layers. For UNet, which contains both Transformer and ResNet blocks, we examine three analogous variants: (i) ResNet - selecting a subset of ResNet blocks with the same number of trainable parameters as ControlNet, (ii) Attn - replacing part of the selected ResNet blocks with attention layers, and (iii) FFN - replacing part of the ResNet subset with feed-forward layers. We use Stable Diffusion 2.1 [33] as the UNet-based model and PixArt-α [5] as the DiT-based model. During training, only the selected subset of parameters is fine-tuned. Control signals are injected using a simple condition embedding, implemented as a learnable MLP, and we evaluate controllability using the Canny edge condition.

As shown in Fig. 5, **finetuning FFN layers consistently yields the largest improvements** in adapting to new conditioning signals across both architectures, compared to finetuning attention layers alone. Notably, this strategy consistently outperforms ControlNet in both UNet and DiT backbones across all training stages. We hypothesize that this effect arises because, in a single-flow architecture, the self- and cross-attention modules already align textual prompts with image features, while the **FFN layers mainly transform and refine the combined representations**. In contrast, modifying attention layers changes the global mixing patterns of features, where small parameter updates can lead to disproportionately large and potentially destabilizing effects.

We illustrate the architectural differences between ControlNet and DeltaControl in Fig. 2. Instead of training an additional control branch, DeltaControl achieves controllability by fine-tuning a small subset of parameters in the base model, significantly reducing overhead compared to ControlNet and RepControlNet. As shown in Tab.4, this design requires fewer trainable parameters, leading to lower GPU memory usage and faster training. At inference time, DeltaControl runs a single-branch that natively incorporates control signals, eliminating the need for a separate control branch. The selected parameters can also be adapted with LoRA, preserving the original model's generative capability. Overall, this design improves both training and inference efficiency. In addition, we introduce AutoAdapter, which enables effective controllable generation.

4.3.2 AutoAdapter.

A direct addition of raw control embeddings to the model's hidden states is fragile in the multi-condition setting: edge, depth, segmentation, and pose signals differ substantially in scale, sparsity, and reliability. These differences can cause either vanishing control signals or over-dominant perturbations. To address this, we introduce **AutoAdapter**, a lightweight module that converts incoming control tokens into hidden-state-compatible residuals prior to backbone injection. Its primary role is twofold: it improves the quality of DeltaControl under heterogeneous conditions and it provides a stable control interface for the alternating student-teacher updates during distillation.

Given a control image c_{in}, we encode it using the base model's VAE encoder to obtain latent control tokens c_0. AutoAdapter then sequentially applies a *distribution matching* module and a *token-wise reliability gate*. The distribution matching module aligns the first- and second-order statistics of the

Table 1: **Quantitative comparison** across six control tasks regarding to FID, Rec, Prec on COCO-5k. "-": the model does not support that condition.

Model	Backbone	NFEs	Canny			Depth			Seg			Pose			Hough			Inpainting		
			FID↓	Rec↑	Prec↑	FID↓	Rec↑	Prec↑	FID↓	Rec↑	Prec↑	FID↓	Rec↑	Prec↑	FID↓	Rec↑	Prec↑	FID↓	Rec↑	Prec↑
ControlNet [51]	SD1.5	50	19.86	0.72	0.65	30.04	0.53	0.42	29.18	0.53	0.52	28.45	0.49	0.56	30.75	0.53	0.49	-	-	-
UniControlNet [52]	SD1.5	50	17.79	0.63	0.65	21.2	0.57	0.54	23.08	0.47	0.49	27.85	0.36	0.49	31.36	0.46	0.44	-	-	-
Uni-Control [31]	SD1.5	50	22.90	0.72	0.59	21.3	0.60	0.50	25.50	0.51	0.46	27.4	0.43	0.43	-	-	-	19.85	0.79	0.75
T2I adapter [27]	SD1.5	50	17.59	0.65	0.56	20.73	0.55	0.54	-	-	-	25.62	0.42	0.48	-	-	-	-	-	-
AnyControl [43]	SD1.5	50	18.55	0.59	0.62	19.32	0.55	0.57	-	-	-	24.99	0.38	0.56	-	-	-	-	-	-
ControlNet++ [17]	SD1.5	50	21.01	0.757	0.726	19.18	0.718	0.641	48.15	0.438	0.478	-	-	-	-	-	-	-	-	-
CTRLORA [47]	SD1.5	50	16.65	0.70	0.55	19.34	0.57	0.50	21.13	0.49	0.48	25.18	0.42	0.46	-	-	-	-	-	-
LCM-ControlNet [6]	SD1.5	4	25.83	0.71	0.65	28.11	0.41	0.50	31.55	0.43	0.50	37.19	0.33	0.56	38.71	0.41	0.49	-	-	-
FlashSD [4]	SD1.5	4	20.42	0.71	0.65	25.49	0.41	0.50	22.75	0.43	0.50	37.19	0.33	0.56	38.71	0.41	0.49	-	-	-
Ctrl-Adapter [18]	SD1.5	8	20.32	0.62	0.70	18.51	0.61	0.75	21.15	0.54	0.71	-	-	-	-	-	-	-	-	-
ControlNet-XS [50]	SD2.1	50	16.79	0.65	0.60	20.11	0.55	0.52	-	-	-	-	-	-	-	-	-	-	-	-
JDM	SD2.1	1	20.62	0.46	0.41	25.06	0.40	0.38	24.95	0.28	0.27	32.84	0.30	0.27	30.65	0.33	0.25	20.73	0.37	0.34
SDXS	SD2.1	1	22.28	0.35	0.29	26.57	0.36	0.32	26.93	0.31	0.26	33.93	0.32	0.29	30.84	0.30	0.24	21.05	0.33	0.31
UNet-based DeltaControl	SD2.1	50	16.22	0.73	0.66	20.06	0.57	0.55	21.64	0.48	0.53	28.98	0.36	0.50	26.48	0.43	0.50	16.41	0.57	0.58
UNet-based FlashControl	SD2.1	1	17.82	0.73	0.63	22.10	0.58	0.55	22.81	0.47	0.52	29.41	0.32	0.53	26.63	0.37	0.52	17.60	0.59	0.59
RepControlNet [9]	PixArt-α	50	16.02	0.81	0.83	20.50	0.62	0.71	22.84	0.55	0.66	28.62	0.42	0.60	26.67	0.48	0.60	15.50	0.75	0.84
DiT-based DeltaControl	PixArt-α	50	15.19	0.81	0.84	**18.11**	0.64	0.76	20.42	0.58	0.68	26.54	0.46	0.66	**24.50**	0.51	0.66	14.90	0.73	0.83
DiT-based FlashControl wo LSP loss	PixArt-α	1	17.83	0.80	0.83	20.01	0.64	0.75	22.64	0.54	0.70	26.62	0.42	0.64	28.16	0.47	0.63	18.95	0.73	0.77
DiT-based FlashControl	PixArt-α	1	16.82	0.82	0.86	18.29	0.66	0.77	20.83	0.58	0.70	**25.62**	0.47	0.66	26.84	0.52	0.64	17.68	0.76	0.79
UniCombine [45]	FLUX.1-schnell	8	20.95	0.66	0.84	25.54	0.50	0.78	-	-	-	-	-	-	-	-	-	19.93	0.62	0.86
OminiControl [44]	FLUX.1-schnell	28	20.14	0.76	0.83	21.47	0.61	0.76	-	-	-	-	-	-	-	-	-	15.43	0.76	**0.88**
OminiControl [44]	FLUX.2-Klein-4B	28	20.63	0.73	0.80	22.02	0.59	0.73	23.83	0.53	0.69	31.63	0.37	0.59	27.30	0.46	0.55	15.92	0.77	0.83
DiT-based DeltaControl	FLUX.2-Klein-4B	28	**12.87**	**0.92**	0.89	**16.57**	**0.73**	0.77	**16.56**	**0.70**	0.69	**22.23**	**0.64**	0.58	**20.34**	**0.68**	0.60	**10.16**	**0.96**	**0.91**
DiT-based FlashControl	FLUX.2-Klein-4B	1	13.64	0.88	**0.89**	16.77	0.70	**0.82**	17.83	0.62	**0.76**	23.68	0.49	**0.70**	21.21	0.55	**0.72**	12.18	0.95	0.91

control tokens to the backbone hidden-state scale:

$$c = \frac{c_0 - \mu_c}{\sigma_c}, \qquad s, b = \mathrm{MLP}(h, c, \phi(t)), \qquad c_{\text{align}} = s \odot c + b.$$

where μ_c and σ_c denote the per-channel mean and standard deviation of c_0, h is the current backbone hidden state, and $\phi(t)$ is the timestep embedding.

AutoAdapter is fundamentally distinct from generic feature projectors. Its explicit objective is to minimize the statistical mismatch between the control stream and the backbone stream before fusion. Formally, we define this mismatch as:

$$\Delta_{\text{stat}}(h, c_{\text{align}}) = \|\mu(c_{\text{align}}) - \mu(h)\|_2^2 + \|\sigma(c_{\text{align}}) - \sigma(h)\|_2^2.$$

This whitening and rescaling process is critical because the denoiser is locally highly sensitive to hidden-state perturbations. Under a first-order approximation, if δ denotes the injected control residual, the perturbed prediction behaves as:

$$\epsilon_\theta(h + \delta, t, y) \approx \epsilon_\theta(h, t, y) + J_\theta(h)\delta,$$

where $J_\theta(h)$ is the Jacobian of the denoiser with respect to the hidden state. When δ is poorly scaled, the resulting update becomes erratic. By ensuring δ matches the backbone statistics, the control signal enters a manifold the pretrained model can reliably absorb, facilitating superior control convergence. Following statistical alignment, AutoAdapter applies a token-wise reliability gate to suppress noisy or incompatible condition tokens. For the i-th token, the gate is computed as:

$$g_i = \sigma\Big(\alpha\, \cos(h_i, c_i^{\text{align}}) + \beta\, \mathrm{time}(t) - \gamma\, \mathrm{MLP}(c_i^{\text{align}})\Big), \qquad h_i \leftarrow h_i + g_i\, c_i^{\text{align}}.$$

where σ is the sigmoid function, and α, β, γ are learnable scalars. The first term assesses compatibility between the current hidden token and the aligned condition token; the second term encourages stronger spatial guidance during early denoising steps when the global layout is formed; the third term penalizes unreliable condition tokens. Because $g_i \in (0, 1)$, the injected residual is strictly bounded:

$$\|\delta_i\|_2 = \|g_i c_i^{\text{align}}\|_2 \le \|c_i^{\text{align}}\|_2.$$

This guarantees that the gate does not fabricate new control content, but merely selectively attenuates the already-aligned control signal.

4.4 Stage 2: One-step Controllable Generative Model FlashControl

AutoAdapter in One-Step Distillation. We retain the AutoAdapter module unchanged during Stage 2, reusing it inside the distilled one-step student. By reusing AutoAdapter, FlashControl inherits the normalized and reliability-weighted residuals optimized in Stage 1. Let the one-step student be

denoted as $f_\theta(z \mid y, c)$ and the trainable fake score as ϵ_ψ. In an alternating distillation framework like DMD, the gradient update follows:

$$\nabla_\theta \mathcal{L}_{\text{DMD}} = \mathbb{E}_{t,y,z,c}\left[w(t)\left(\hat{\epsilon}_\phi - \hat{\epsilon}_\psi\right)\frac{\partial f_\theta(z \mid y, c)}{\partial \theta}\right].$$

Without AutoAdapter's alignment, control perturbations fluctuate sharply across timesteps and condition types, escalating the variance of both the student and LoRA fake score updates. By enforcing $\delta_{\text{ctrl}} = g \odot c_{\text{align}}$, the injected residual is normalized before reaching either path. Consequently, AutoAdapter actively supports faster and more stable convergence in distillation.

Latent Sensitivity Preservation (LSP). While score distillation provides an efficient formulation for one-step generation, we observe a distinct optimization instability specific to controllable distillation: the student rapidly becomes overly mode-seeking. It overfits to a narrow teacher trajectory, loses sensitivity to the input Gaussian noise, and collapses to a near-deterministic mapping for any given prompt-condition pair (y, c). Qualitatively, this manifests as late-stage stylized collapse and oversmoothing; quantitatively, it produces sharp, disruptive spikes in both student and LoRA losses during late training.

To mitigate this, we introduce the **Latent Sensitivity Preservation (LSP)** objective. For two i.i.d. noise inputs $\mathbf{z}$ and $\mathbf{z}'$ in $\mathcal{N}(0, I)$, we extract the corresponding image latents produced by the student model when using the same text prompt $\mathbf{y}$ and condition $\mathbf{c}$:

$$\mathbf{x}_0 \;=\; f_\theta(\mathbf{z} \mid \mathbf{y}, \mathbf{c}), \qquad \mathbf{x}_0' \;=\; f_\theta(\mathbf{z}' \mid \mathbf{y}, \mathbf{c}). \tag{1}$$

Then, we compute the output variation $d_{\text{out}} = D(\mathbf{x}_0, \mathbf{x}_0')$ and the input variation $d_{\text{in}} = D(\mathbf{z}, \mathbf{z}')$, given D is a distance metric like ℓ_2. Our **LSP objective** is computed as:

$$\mathcal{L}_{\text{LSP}} \;=\; -\frac{d_{\text{out}}}{d_{\text{in}} + \varepsilon} \tag{2}$$

with ε is machine epsilon to avoid division by zero. This recipe aims to maximize Δoutput / Δinput, discourage the "flat" mapping $f_\theta(\mathbf{z})$, prevent generators from ignoring the latent noise. In one-step distillation, the only stochastic source is $\mathbf{z}$, preserving a non-trivial Jacobian $\frac{\partial f_\theta}{\partial z}$ is critical for **diversity and sharpness** [24].

Crucially, LSP is a bounding regularization rather than an unbounded diversity maximizer. It preserves conditional faithfulness while strictly preventing degenerate one-to-one mapping collapse. The final FlashControl objective is formulated as:

$$\mathcal{L}_{\text{FlashControl}} = \mathcal{L}_{\text{DMD}} + \lambda_{\text{LSP}}\mathcal{L}_{\text{LSP}},$$

where $\mathcal{L}_{\text{DMD}}$ is the baseline distillation objective and λ_{LSP} scales the sensitivity regularization. In summary, our complete pipeline achieves state-of-the-art one-step fidelity because AutoAdapter transforms the control signal into an easily optimizable state, while LSP guarantees the student does not undergo deterministic collapse once that control signal dominates the generative trajectory.

5 Experiments

5.1 Experimental Setup

Training details. We provide complete implementation detail in Appendix Section F.

Evaluation protocol. Follow previous work, we leverage COCO2017-5k [19] for evaluation dataset. We evaluate six control types (Canny, Depth, Segmentation, Pose, Hough lines, and Inpainting) for both multi-step and one-step models (1 NFE) under same setting using three metrics FID[13], Precision (Prec) and Recall (Rec).

Efficiency of ControlNet vs DeltaControl. We quantified the training and inference efficiency of a dual-branch ControlNet against our single-branch **DeltaControl**. First, we set up training with the same protocol for both models, with backbone PixArt-α with a single condition. We measure inference time of ControlNet, DeltaControl, and FlashControl in Table 3.

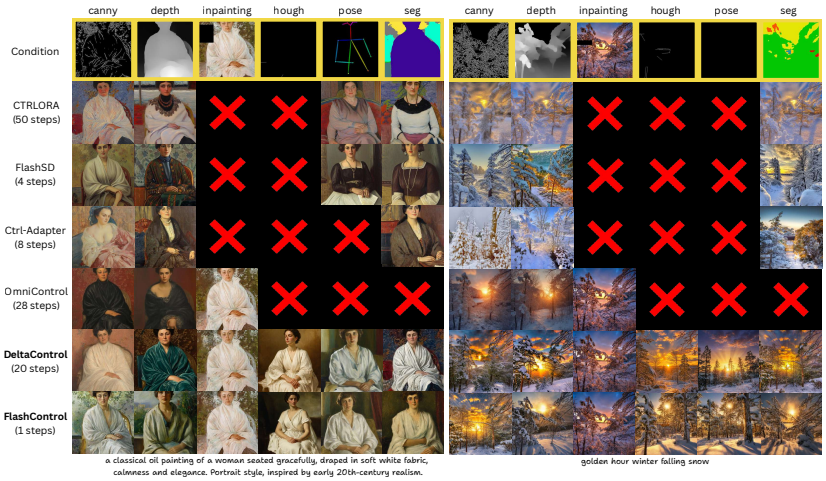


Figure 3: Qualitative comparison across control modalities including: Canny, Depth, Inpainting, Hough, Pose, and Segmentation. Crossed cells mark unsupported controls.

Table 2: **Ablation on AutoAdapter:** FID↓ on 30k iters.

Method	Canny	Depth	Seg
– AutoAdapter	28.15	31.11	31.61
– distribution match	26.38	29.51	30.70
– token-wise gate	26.88	29.64	31.19
+ **AutoAdapter**	**26.03**	**29.24**	**30.62**

Table 3: **Inference time per image.**

Method	s/img
ControlNet	2.1748
DeltaControl	2.0576
FlashControl	0.0429

Table 4: **Training efficiency** under the same protocol with SD2.1.

Method	Fwd	Bwd	Mem	Train.	Total
ControlNet	0.2689	0.1450	30GB	294MB	904MB
T2I-Adapter	0.2486	0.0601	22GB	83MB	693MB
RepControlNet	0.2710	0.0970	18.8GB	297MB	908MB
DeltaControl	0.2411	0.0714	17GB	297MB	610MB

5.2 Quantitative comparison

Table 1 compares our multi-step teacher **DeltaControl** and one-step student **FlashControl** against prior controllable baselines across all control types. We can observe that across tasks, our DiT-based one-step model matches or surpasses prior multi-step baselines. For example, on Canny, FlashControl (FLUX.2-Klein-4B, 1 NFE) achieves **FID 13.64**, essentially closing the gap to its 28-step teacher 12.70, while outperforming strong multi-step baselines like OminiControl 20.14. The same scenario happens for other conditionings. This indicates that the distillation preserves fidelity and coverage. We also surpass other controllable one-step baselines like JDM [23] and SDXS [42] too.

5.3 Qualitative comparison

We compare **DeltaControl** and **FlashControl** (PixArt-α) with recent controllable generators: CTRLORA, FlashSD, Ctrl-Adapter and OminiControl. We evaluate six common conditioning types. For each method we use its recommended sampling process. In Fig. 3, for the portrait, FlashControl keeps consistent brushwork and fabric detail. For outdoor scenes, it maintains coherent lighting and snow particle density, avoiding over-sharpening typical of aggressive control. DeltaControl and FlashControl produce aligned image with high perceptual quality, whereas several baselines introduce color shifts around the hole boundary. Despite using 1 step, FlashControl delivers quality on par with multi-step methods (e.g., 28–50 steps), demonstrating controllability in real time. Additionally, we also provide control fidelity benchmarking in Appendix.

5.4 Ablation study

AutoAdapter ablation. We study AutoAdapter on 3 conditions (Canny, Depth, Seg), comparing: (i) no adapter, (ii) w/o distribution matching, (iii) w/o token-wise gate, and (iv) full AutoAdapter (Tab 2). Full AutoAdapter gives the lowest FID and fastest convergence for all conditions. Removing either component degrades FID, and removing the adapter entirely yields the worst and less stable training. We also study how AutoAdapter make distillation converge better in Section A.

Table 5: Effect of teacher architecture under the same one-step distillation pipeline. All rows use the same distillation recipe; only the teacher architecture changes. Metric is FID on COCO-5k.

Teacher Architecture	Backbone	Canny ↓	Depth ↓	Seg ↓	Pose ↓	Hough ↓	Inpainting ↓
ControlNet	SD2.1	47.15	49.12	52.62	58.03	53.01	31.74
T2I-Adapter	SD2.1	24.72	29.62	28.51	37.80	33.72	24.15
RepControlNet	SD2.1	21.41	25.29	26.40	34.33	31.01	21.23
DeltaControl (Ours)	SD2.1	**19.53**	**24.27**	**24.93**	**32.62**	**29.13**	**19.07**
OminiControl	FLUX.2-Klein-4B	17.38	23.01	25.42	32.51	30.55	17.40
DeltaControl (Ours)	FLUX.2-Klein-4B	**11.27**	**16.95**	**24.63**	**30.91**	**29.45**	**15.94**

LSP in distillation. We ablate the LSP loss in FlashControl to test diversity/sharpness. Without it, FID worsens (Tab. 1), suggesting the loss helps avoid near-deterministic collapse and keeps the one-step model sensitive to input noise. We also compare dual-branch ControlNet vs. our single-branch **DeltaControl** under the same PixArt-α setup (single condition): DeltaControl reduces backward time by 50% and training memory by 40% (Tab. 4). Finally, FlashControl is fastest at inference, achieving 0.042s per 512×512 image on an NVIDIA A100 40GB (bfloat16) (Tab Table 2). Additionally, we examine how LSP fight collapse in distillation in Section B.

Varying Teacher Architecture. Tab. 5 isolates a single factor: the teacher architecture. All rows use the same one-step distillation pipeline with varying teacher model. This comparison shows that the improvement does not come from the distillation recipe alone. Branch-based teachers such as ControlNet and T2I-Adapter transfer poorly, confirming discussions in Section 4.3. More importantly, even among *single-stream* teachers, **DeltaControl consistently yields the strongest one-step student**. This shows that single-stream design by itself is insufficient: the teacher must be structured specifically for distillation. RepControlNet improves inference-time structure through reparameterization, and OminiControl provides a minimal universal controller for DiT backbones, but DeltaControl is the one that exposes the cleanest teacher-to-student distillation interface.

5.5 Extended Alignment Metrics for Quantitative Results

In addition to the primary alignment metrics presented in the main paper including FID, Precision, and Recall, we also include task-specific metrics to more clearly evaluate how well the model adheres to different conditioning inputs. Follow [43], we evaluate structural consistency using RMSE for both canny edge and depth conditions, and measure semantic alignment for segmentation conditions using mean Pixel Accuracy (mPA).

Table 6: Comparison of model performance on COCO-5K using extended alignment metrics. "-" denotes that condition is unavailable.

Models	Backbone	Canny (RMSE ↓)	Depth (RMSE ↓)	Seg (mPA ↑)
ControlNet	SD1.5	87.30	36.49	0.12
CTRLORA	SD1.5	82.52	33.59	0.16
CtrlAdapter	SD1.5	84.05	38.43	0.13
DeltaControl	SD2.1	76.25	34.75	0.16
FlashControl	SD2.1	76.64	34.12	0.18
RepControlNet	PixArt-α	87.56	32.77	0.19
DeltaControl	PixArt-α	76.63	31.30	0.19
FlashControl	PixArt-α	75.32	**28.63**	**0.20**
UniCombine	FLUX.1-schnell	75.35	37.30	-
OminiControl	FLUX.2-Klein-4B	**74.83**	32.74	0.16
DeltaControl	FLUX.2-Klein-4B	75.78	30.72	0.22
FlashControl	FLUX.2-Klein-4B	76.84	**29.24**	**0.23**

6 Conclusion

FlashControl shows that single-step, controllable image generation is possible with a streamlined, single-branch design. It preserves much of the teacher's controllability while simplifying both training and inference. In evaluations, it performs on par with traditional two-branch methods. For future work, we can investigate simultaneous multi-condition composition (e.g., combining depth and pose in a single input).

Acknowledgments and Disclosure of Funding

This work was supported by Qualcomm AI Research. The authors declare no additional competing interests beyond their disclosed affiliations.

References

[1] O. Avrahami, D. Lischinski, and O. Fried. Blended diffusion for text-driven editing of natural images. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 18187–18197. IEEE, June 2022. doi: 10.1109/cvpr52688.2022.01767. URL http://dx.doi.org/10.1109/CVPR52688.2022.01767.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[3] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[4] C. Chadebec, O. Tasar, E. Benaroche, and B. Aubin. Flash diffusion: Accelerating any conditional diffusion model for few steps image generation, 2024.

[5] J. Chen, J. Yu, C. Ge, L. Yao, E. Xie, Y. Wu, Z. Wang, J. Kwok, P. Luo, H. Lu, and Z. Li. Pixart-alpha: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

[6] J. Chen, Y. Wu, S. Luo, E. Xie, S. Paul, P. Luo, H. Zhao, and Z. Li. Pixart-delta: Fast and controllable image generation with latent consistency models, 2024.

[7] J. Chen, S. Xue, Y. Zhao, J. Yu, S. Paul, J. Chen, H. Cai, S. Han, and E. Xie. Sana-sprint: One-step diffusion with continuous-time consistency distillation, 2025. URL https://arxiv.org/abs/2503.09641.

[8] T. Dao, T. H. Nguyen, T. Le, D. Vu, K. Nguyen, C. Pham, and A. Tran. Swiftbrush v2: Make your one-step diffusion model better than its teacher, 2024. URL https://arxiv.org/abs/2408.14176.

[9] Z. Deng, K. Zhou, F. Wang, and Z. Mi. Repcontrolnet: Controlnet reparameterization, 2024. URL https://arxiv.org/abs/2408.09240.

[10] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn, Z. English, K. Lacey, A. Goodwin, Y. Marek, and R. Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. URL https://arxiv.org/abs/2403.03206.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

[12] G. Gu, B. Ko, S. Go, S.-H. Lee, J. Lee, and M. Shin. Towards light-weight and real-time line segment detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[13] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. URL https://arxiv.org/abs/1706.08500.

[14] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[15] B. F. Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

[16] B. F. Labs, S. Batifol, A. Blattmann, F. Boesel, S. Consul, C. Diagne, T. Dockhorn, J. English, Z. English, P. Esser, S. Kulal, K. Lacey, Y. Levi, C. Li, D. Lorenz, J. Müller, D. Podell, R. Rombach, H. Saini, A. Sauer, and L. Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space, 2025. URL https://arxiv.org/abs/2506.15742.

[17] M. Li, T. Yang, H. Kuang, J. Wu, Z. Wang, X. Xiao, and C. Chen. Controlnet++: Improving conditional controls with efficient consistency feedback. In *European Conference on Computer Vision*, 2024.

[18] H. Lin, J. Cho, A. Zala, and M. Bansal. Ctrl-adapter: An efficient and versatile framework for adapting diverse controls to any diffusion model. *arXiv preprint arXiv:2404.09967*, 2024.

[19] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312.

[20] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. URL https://arxiv.org/abs/2209.03003.

[21] C. Lu, Y. Zhou, F. Bao, J. Chen, C. Li, and J. Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022. URL https://arxiv.org/abs/2206.00927.

[22] S. Luo, Y. Tan, L. Huang, J. Li, and H. Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference, 2023. URL https://arxiv.org/abs/2310.04378.

[23] Y. Luo, T. Hu, Y. Song, J. Sun, Z. Li, and J. Tang. Adding additional control to one-step diffusion with joint distribution matching, 2025. URL https://arxiv.org/abs/2503.06652.

[24] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang. Mode seeking generative adversarial networks for diverse image synthesis, 2019. URL https://arxiv.org/abs/1903.05628.

[25] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations, 2022. URL https://arxiv.org/abs/2108.01073.

[26] C. Meng, R. Rombach, R. Gao, D. P. Kingma, S. Ermon, J. Ho, and T. Salimans. On distillation of guided diffusion models, 2023. URL https://arxiv.org/abs/2210.03142.

[27] C. Mou, X. Wang, L. Xie, Y. Wu, J. Zhang, Z. Qi, Y. Shan, and X. Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[28] T. H. Nguyen and A. Tran. Swiftbrush: One-step text-to-image diffusion model with variational score distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7807–7816, 2024.

[29] V. Nguyen, A. Nguyen, T. Dao, K. Nguyen, C. Pham, T. Tran, and A. Tran. Supercharged one-step text-to-image diffusion models with negative prompts, 2025. URL https://arxiv.org/abs/2412.02687.

[30] W. Peebles and S. Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

[31] C. Qin, S. Zhang, N. Yu, Y. Feng, X. Yang, Y. Zhou, H. Wang, J. C. Niebles, C. Xiong, S. Savarese, et al. Unicontrol: A unified diffusion model for controllable visual generation in the wild. *arXiv preprint arXiv:2305.11147*, 2023.

[32] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1623–1637, 2020.

[33] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[34] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2022. URL https://arxiv.org/abs/2112.10752.

[35] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, 2023. URL https://arxiv.org/abs/2208.12242.

[36] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models, 2022. URL https://arxiv.org/abs/2202.00512.

[37] A. Sauer, D. Lorenz, A. Blattmann, and R. Rombach. Adversarial diffusion distillation, 2023. URL https://arxiv.org/abs/2311.17042.

[38] A. Sauer, F. Boesel, T. Dockhorn, A. Blattmann, P. Esser, and R. Rombach. Fast high-resolution image synthesis with latent adversarial diffusion distillation, 2024. URL https://arxiv.org/abs/2403.12015.

[39] C. Schuhmann, R. Beaumont, R. Vencu, C. W. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. R. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[40] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

[41] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models, 2023. URL https://arxiv.org/abs/2303.01469.

[42] Y. Song, Z. Sun, and X. Yin. Sdxs: Real-time one-step latent diffusion models with image conditions, 2024. URL https://arxiv.org/abs/2403.16627.

[43] Y. Sun, Y. Liu, Y. Tang, W. Pei, and K. Chen. Anycontrol: Create your artwork with versatile control on text-to-image generation, 2024.

[44] Z. Tan, S. Liu, X. Yang, Q. Xue, and X. Wang. Ominicontrol: Minimal and universal control for diffusion transformer. 2025.

[45] H. Wang, J. Peng, Q. He, H. Yang, Y. Jin, J. Wu, X. Hu, Y. Pan, Z. Gan, M. Chi, et al. Unicombine: Unified multi-conditional combination with diffusion transformer. *arXiv preprint arXiv:2503.09277*, 2025.

[46] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[47] Y. Xu, Z. He, S. Shan, and X. Chen. Ctrlora: An extensible and efficient framework for controllable image generation. *arXiv preprint arXiv:2410.09400*, 2024.

[48] T. Yin, M. Gharbi, T. Park, R. Zhang, E. Shechtman, F. Durand, and W. T. Freeman. Improved distribution matching distillation for fast image synthesis. In *NeurIPS*, 2024.

[49] T. Yin, M. Gharbi, R. Zhang, E. Shechtman, F. Durand, W. T. Freeman, and T. Park. One-step diffusion with distribution matching distillation. In *CVPR*, 2024.

[50] D. Zavadski, J.-F. Feiden, and C. Rother. Controlnet-xs: Rethinking the control of text-to-image diffusion models as feedback-control systems. In *European Conference on Computer Vision*, pages 343–362, 2024.

[51] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[52] S. Zhao, D. Chen, Y.-C. Chen, J. Bao, S. Hao, L. Yuan, and K.-Y. K. Wong. Uni-controlnet: All-in-one control to text-to-image diffusion models. *Advances in Neural InformationProcessing Systems*, 2023.

[53] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 633–641, 2017.

Appendix Overview. This appendix is organized as follows. We first provide extended quantitative alignment metrics for Canny, depth, and segmentation conditions in Table 6. We then analyze the training behavior of one-step distillation, including the benefits of AutoAdapter in Section A, the stabilizing effect of LSP in Section B, and the feature-alignment analysis in Fig. 8. Next, we present the user study in Section , followed by related work in Section 2, dataset construction, and training implementation details in Section F. We further study the effect of the number of training conditions in Fig. 10 and compare different distillation objectives in Table 8. Finally, we provide additional experimental details in Section I, discuss limitations in Section J, describe AutoAdapter in detail in Section L, and show more qualitative results in Figs. 11 to 16.

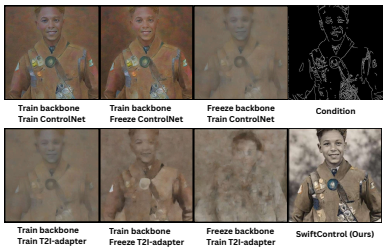
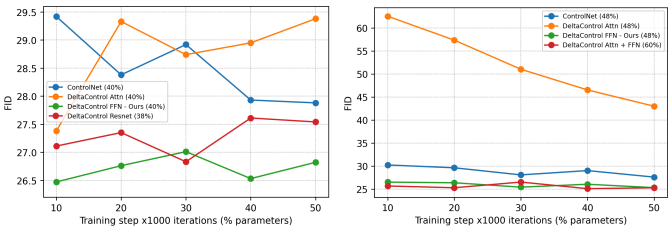


Figure 4: One-step distillation with ControlNet, T2I-Adapter vs Delta-Control.



(a) UNet (SD2.1) (b) DiT (PixArt-α)

Figure 5: Effect of selective fine-tuning on convergence (lower is better)

A AutoAdapter benefits distillation.

We investigate whether AutoAdapter actively benefits the one-step distillation process, rather than merely strengthening the Stage-1 teacher. We compare the optimization trajectories of the plain single-stream baseline, `SS + AutoAdapter`, and `SS + AA + LSP` under identical distillation conditions. Visually, AutoAdapter establishes recognizable structure much earlier in training and yields consistently cleaner conditional outputs Fig. 7. Quantitatively, it produces lower and substantially smoother loss curves prior to the late-stage instability region Fig. 6. These findings confirm that AutoAdapter transcends simple Stage-1 quality improvements: by providing a statistically well-conditioned control interface, it makes the alternating Stage-2 updates inherently easier and more stable to optimize.

B LSP benefits distillation.

We examine the training dynamics to isolate the stabilizing effect of LSP. As shown in 6, distillation without LSP exhibits a pronounced spike in both student and LoRA losses. This quantitative instability manifests visually in 6, where the ablated model drifts into severe degradation and oversaturationin training. Conversely, the inclusion of LSP yields flat, stable loss curves and preserves coherent visual structure throughout the same training range. These findings substantiate our core theoretical claim: LSP functions strictly as a collapse-prevention mechanism. Rather than acting as a generic diversity heuristic, it actively preserves the student's sensitivity to input noise, successfully preventing the model from degenerating into a near-deterministic mapping once the spatial control signal dominates.

C Effect of AutoAdapter on feature alignment.

We further analyze AutoAdapter by tracking the statistic mismatch Δ_{stat} during training. As shown in Fig. 8, all variants begin from a similar starting point, but the full AutoAdapter reduces the mismatch more quickly and remains consistently lowest throughout training. Removing AutoAdapter gives the highest Δ_{stat} over nearly the whole trajectory, showing that naive condition injection leaves a persistent mismatch between the control features and the backbone hidden states. Removing either distribution matching or the token-wise gate leads to intermediate behavior: both variants improve over no adapter, but both remain worse than the full model. Overall, this result supports our claim that AutoAdapter improves controllable generation by making the injected control features more

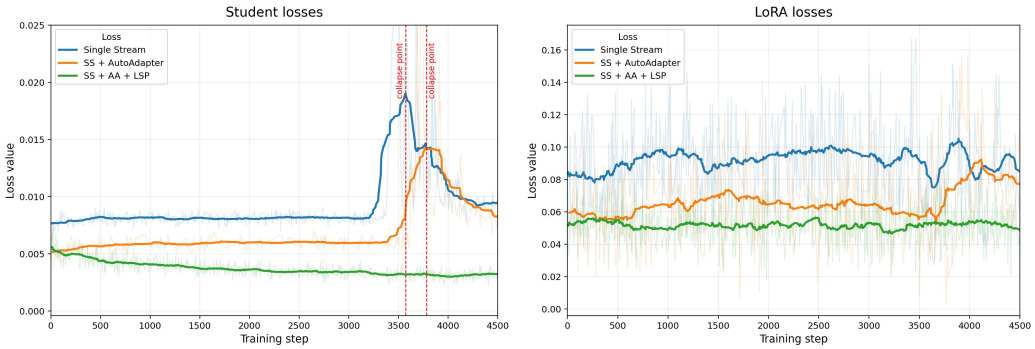


Figure 6: Training losses during one-step distillation. We compare *Single stream*, *SS (Single Stream) + AutoAdapter*, and *SS + AA (AutoAdapter) + LSP* using both student loss and LoRA-teacher loss across training steps. AutoAdapter makes optimization smoother and lowers the losses compared with the plain single-stream baseline, while adding LSP further suppresses the loss spike around 3.5k–4k steps, indicating more stable distillation.



Figure 7: Qualitative training at different distillation iterations. AutoAdapter improves visual convergence over the plain single-stream baseline, and adding LSP further stabilizes late-stage training and avoids collapse. SS is Single-Stream and AA is AutoAdapter.

compatible with the backbone feature space. All experiments are employed with the same backbone SD2.1, loss DMD only and hyperparameters to ensure fairness.

D User Study

We conducted a user study with 279 participants to evaluate perceptual quality. Each participant compared outputs from four methods (FlashControl, ControlNet, Ctrl-Adapter, CtrLoRA) under the same condition and prompt, selecting the best image based on Image Quality, Condition Alignment, and Textual Alignment. We used 30 prompts (10 each for Canny, Depth, and Segmentation). As shown in Section , FlashControl was preferred in **57.3%** of cases for Image Quality, **43.0%** for Condition Alignment, and **47.0%** for Textual Alignment, outperforming all baselines despite using only **1 NFE** versus 50 for others. This shows that our one-step method achieves superior perceptual quality while being $50\times$ faster.

Instant Controllable Diffusion Models. Several recent works share the same goal as ours: one-step controllable diffusion generation. SDXS [42] distills a controllable student from a multi-step teacher, but still shows a noticeable quality and diversity gap compared to strong multi-step controllable

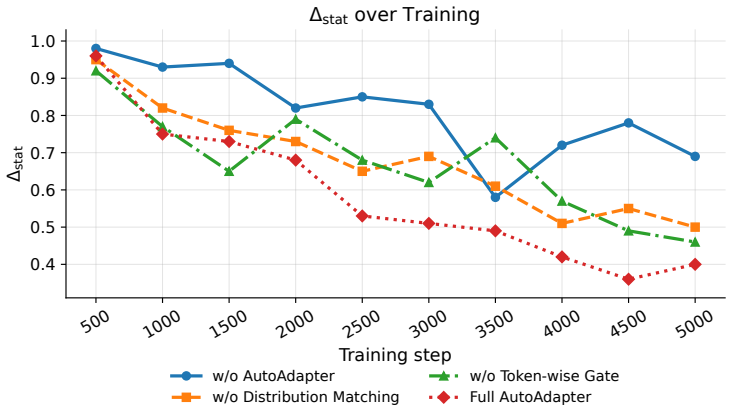


Figure 8: Evolution of the control-hidden statistic mismatch Δ_{stat} during training. All variants start from a similar level, but the full AutoAdapter reduces the mismatch fastest and maintains the lowest Δ_{stat} throughout training. Removing AutoAdapter leads to the highest mismatch, while removing either distribution matching or the token-wise gate gives intermediate performance.

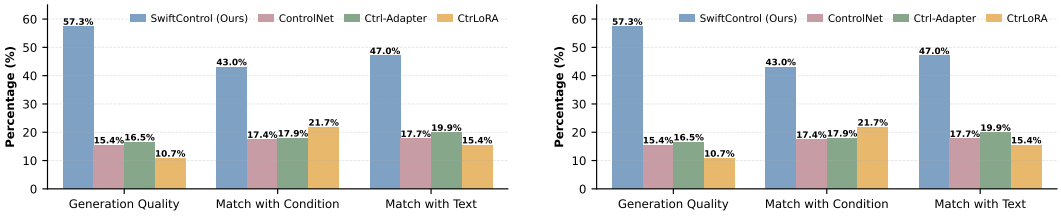


Figure 9: User study. Our FlashControl achieves higher perceptual quality and $50\times$ faster.

models. JDM [23] provides a principled joint distribution matching objective over images and conditions, but its training pipeline is more complex, has been mainly examined with a single UNet backbone, and does not directly address the teacher-architecture mismatch that arises when compressing controllable generation into one step. Additionally, neither method has a publicly available training implementation.

E Dataset.

To train a controllable DeltaControl, we use a 2M subset of the Re-LAION5B dataset [39] as supervised images for consistency with other baseline works. Following ControlNet [51], we extract corresponding conditions from the dataset, including Canny Edge [2], Depth Map [32], Segmentation Mask [53], M-LSD lines [12], Openpose [3] and Inpainting. We extract all control modalities from the same LAION images by following the ControlNet preprocessing pipeline on their official GitHub. Some prior works use different condition extraction pipelines (e.g., similar canny settings but different depth estimators), so we follow the ControlNet to unify condition generation across all modalities. For FlashControl distillation, we use only caption and conditions since no image is required.

F Training Implementation Details

We summarize the implementation details for Stage 1 **DeltaControl** and Stage 2 **FlashControl** in Table 7 across three backbones Stable Diffusion 2.1, PixArt-α and FLUX.2-klein-4B. For training LoRA layers, we apply LoRA to all FFN modules described in Section I for Stable Diffusion 2.1 and PixArt-α. For FLUX.2-klein-4B, we follow the same principle: LoRA is applied to all FFN layers in the double-stream blocks and to the first 14 fused projection layers in the single-stream blocks, so that the total number of trainable parameters remains comparable to the ControlNet setup.

Table 7: Training hyperparameters and compute settings for Stage 1 **DeltaControl** and Stage 2 **FlashControl**. For the conditioning projectors, we randomly initialize these weights and also train them using LoRA.

Stage 1 **DeltaControl**			
Backbone	Stable Diffusion 2.1	PixArt-α	FLUX.2-klein-4B
# of training data	2M supervised images		
Condition types	Canny, Depth, Segmentation, Hough, Pose, Inpainting		
# of LoRA parameters	412M	370M	190M
# of AutoAdapter parameters	4M	4M	30M
# Total parameter	614M	864M	3.9B
LoRA rank	512	512	64
Learning rate	$\epsilon = 5 * 10^{-5}$		
Optimizer	Adam ($\beta_1 = 0, \beta_2 = 0.999, \epsilon = 10^{-8}$)		
Resolution	512		
# of GPUs	8×H100 (80G)		
Batch size per GPU	16		
Training Time (Hours)	3		
Seed	42		
Stage 2 **FlashControl**			
Backbone	Stable Diffusion 2.1	PixArt-α	FLUX.2-klein-4B
Paradigm	Diffusion	Diffusion	Rectified Flow
Fake score LoRA rank	64	64	128
Student LoRA rank	256	256	128
Resolution	512		
Teacher Guidance CFG	3		
Optimizer	Adam ($\beta_1 = 0, \beta_2 = 0.999, \epsilon = 10^{-8}$)		
λ_{LSP}	0.1		
Diversity Distance	Cosine		
# of GPUs	8×H100 (80G)		
Batch size per GPU	8		
Training Time (Hours)	4		
Seed	42		

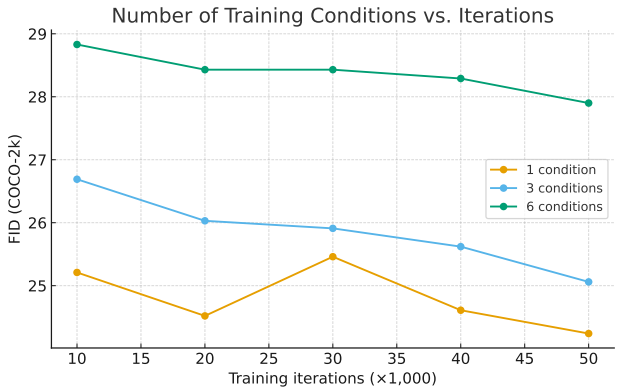


Figure 10: **Training Difficulty**. With more control types and under the same budget, FID and convergence rate are slightly worse.

G Number of Training Conditions

To see whether training with more control types degrades general quality, we vary the number of condition types seen during training (1, 3, and 6) and track FID during training with the same evaluation protocol, as in Fig. 10. The result is that, with the same model size and training budget, more conditions hurt image quality. This may be because giving the model more condition types forces its parameters to explain a broader set of inputs.

Table 8: Quantitative results (FID) on different distillation methods across 6 tasks. FlashControl leverages DMD.

	Canny	Depth	Seg	Pose	Hough	Inpainting
DMD PixArt-α (**Ours**)	17.83	20.10	22.64	26.62	28.16	18.95
DMD2	16.45	19.29	21.29	26.38	28.95	19.02

H Varying Distillation Methods

To study how the choice of distillation method affects our framework, we replace DMD with DMD2. Both configs have LSD loss. With adversarial loss, DMD2 bring more robust performance compared with DMD.

I More Details About Our Experiments

Selective parameter fine-tuning experiment. In Fig. 5, we compare ControlNet with our DeltaControl variants on both a U-Net–based model (Stable Diffusion 2.1) and a DiT-based model (PixArt-α).

PixArt-α has 28 transformer blocks; for each DeltaControl variant:

- **Attn**: fine-tune all parameters in the self-attention and cross-attention layers of all 28 blocks.
- **FFN**: fine-tune all feedforward layers in all 28 blocks.
- **Attn+FFN**: fine-tune a subset of parameters taken from both the Attn and FFN variants, again across all 28 blocks.

Stable Diffusion 2.1 has 4 down blocks, 1 mid block, and 4 up blocks; for each DeltaControl variant:

- **ResNet**: fine-tune all convolution layers in the ResNet sub-blocks of all down blocks, the mid block, and the first two up blocks.
- **FFN**: fine-tune all convolution layers in the ResNet sub-blocks of all down blocks and the mid block, plus all feedforward layers in the transformer blocks.
- **Attn**: fine-tune all convolution layers in the ResNet sub-blocks of all down blocks, the mid block, and the first up block, plus all self-attention and cross-attention layers in the transformer blocks.

We choose the number of tuned layers so that the percentage of trainable parameters matches ControlNet for a fair comparison. All models are trained with a single Canny edge condition: we (i) encode the condition with a VAE, (ii) map it with a small trainable embedding module, and (iii) add it to the intermediate latent at the network head in the same way for all variants.

J Limitations

FlashControl has several limitations. First, although the final student is fast at inference, the method still requires a strong pretrained backbone and a two-stage teacher–student training pipeline, which introduces non-trivial training cost. Second, our experiments focus on 512×512 images and six common spatial controls, so we do not claim direct generalization to higher-resolution synthesis, video, or specialized domains. Third, FlashControl currently handles one control type at a time. Our results suggest that training with more condition types under the same budget can slow convergence and slightly degrade quality, so robust multi-condition composition remains future work. Finally, like other controllable image generators, fast one-step inference may lower the cost of creating misleading visual content; responsible release should include usage restrictions, provenance tools, and safety filters.

K More Qualitative Results

More qualitative results for DeltaControl and FlashControl are shown in Figs. 11 to 16.

L AutoAdapter: Detailed Design

In this section, we provide a more detailed explanation of the *AutoAdapter* module introduced in Section 4.2 of the main paper. Recall that AutoAdapter is a lightweight head that converts a spatial control input (*e.g.*, a Canny edge map or a depth map) into token features that can be safely injected into the single-flow backbone of DeltaControl.

Overall data flow. Given a single control image c_{in} and backbone hidden states $h \in \mathbb{R}^{B\times N\times D}$ at the network head, AutoAdapter proceeds in three high-level stages:

1. Encode the condition image into a latent embedding c_0 using the same VAE encoder as the base diffusion model, then project it to the same dimension as h.
2. Match the distribution of the condition tokens to the hidden-state tokens using the *distribution matching module* (whitening and re-scaling, as defined in Eqs. (5)–(7) of the main paper).
3. Modulate each condition token with a *token-wise uncertainty gate*, and add the gated condition features to the backbone hidden states.

This design lets the model interpret control inputs in the same feature space as the backbone, while automatically deciding *where* and *how strongly* each control token should affect the current latent. We provide pseudo-code for the AutoAdapter module in Algorithm 1.

Distribution matching in practice. We (i) VAE-encode c_{in} to c_0, (ii) whiten per-channel over tokens as in Eq. (5), (iii) feed the whitened tokens, the current hidden-state tokens, and the timestep embedding $\phi(t)$ into a small MLP to predict per-channel scale and shift parameters (s, b) as in Eq. (6), and (iv) re-scale and shift to produce c_{align} as in Eq. (7).

Intuitively, this module performs an AdaIN-style alignment between the condition stream and the backbone stream. Without this step, directly adding raw condition tokens to h can either drown out the original features (if the condition is too large) or make the control nearly invisible (if it is too small). The distribution matching module keeps the condition in a "safe" range where gradients remain stable across timesteps and conditions.

Token-wise uncertainty gating. Even after distribution matching, not every condition token should affect the hidden state equally. For example, a poorly extracted edge map or an occluded region should be trusted less than a clean, well-aligned region. AutoAdapter therefore applies a token-wise gate g_i to each condition token $c_{\text{align},i}$ before adding it to the corresponding hidden token h_i.

The gate g_i is computed from three signals:

1. A similarity term between the hidden token and the aligned condition token (high similarity $\Rightarrow$ gate opens).
2. A time-dependent factor time(t) that encourages stronger control at early denoising steps and weaker control near the final steps.
3. An uncertainty term produced by a small MLP on $c_{\text{align},i}$, which decreases the gate when the condition token appears noisy or unreliable.

These three terms are combined into a single scalar gate and squashed by a sigmoid, as described in Sec. 4.2.2. The gated condition token is then added to the hidden state:

$$h_i \leftarrow h_i + g_i \cdot c_{\text{align},i}.$$

In regions where the condition aligns well with the current latent (*e.g.*, edges or poses that agree with the generated structure), g_i is close to 1 and the control strongly influences the hidden state. When the condition is inconsistent or noisy, g_i shrinks toward 0, effectively turning the control off in those tokens.

Algorithm 1 AutoAdapter flow

Require: Hidden states $h \in \mathbb{R}^{B\times N\times D}$ at the backbone head, condition image c_{in} (single active condition), timestep t.

Ensure: Updated hidden states $\tilde{h}$ with injected control.

// Encode condition as latent tokens

$c_0 \leftarrow \text{Enc}(c_{\text{in}})$ {Same VAE encoder as the base model}

$c_0 \leftarrow \text{Flatten}(c_0)$ {Shape: $B \times \tilde{N} \times D$}

// Distribution matching

$c \leftarrow \text{WhitenCondition}(c_0)$ {Per-channel whitening as in Eq. (5)}

$\phi_t \leftarrow \text{TimestepEmbedding}(t)$

$(s, b) \leftarrow \text{MLP}_{\text{align}}(h, c, \phi_t)$ {Per-channel scales and shifts as in Eq. (6)}

$c_{\text{align}} \leftarrow \text{RescaleCondition}(c, s, b)$ {Aligned condition tokens as in Eq. (7)}

// Token-wise uncertainty gating

for $i = 1$ **to** N **do**

  $h_i \leftarrow \text{Norm}(h_i)$ {Optional: e.g., ℓ_2-normalization}

  $c_i \leftarrow c_{\text{align},i}$ {Token aligned to hidden index i}

  $\text{sim}_i \leftarrow \text{CosineSimilarity}(h_i, c_i)$

  $\tau_i \leftarrow \text{TimeFactor}(t)$ {Stronger at early timesteps, weaker at late timesteps}

  $u_i \leftarrow \text{MLP}_{\text{uncertainty}}(c_i)$ {Predicts token-wise uncertainty}

  $g_i \leftarrow \text{GateFromSignals}(\text{sim}_i, \tau_i, u_i)$ {Combine signals and apply sigmoid as in Sec. 4.2.2}

  $h_i \leftarrow h_i + g_i \cdot c_i$

end for

$\tilde{h} \leftarrow h$

return $\tilde{h}$

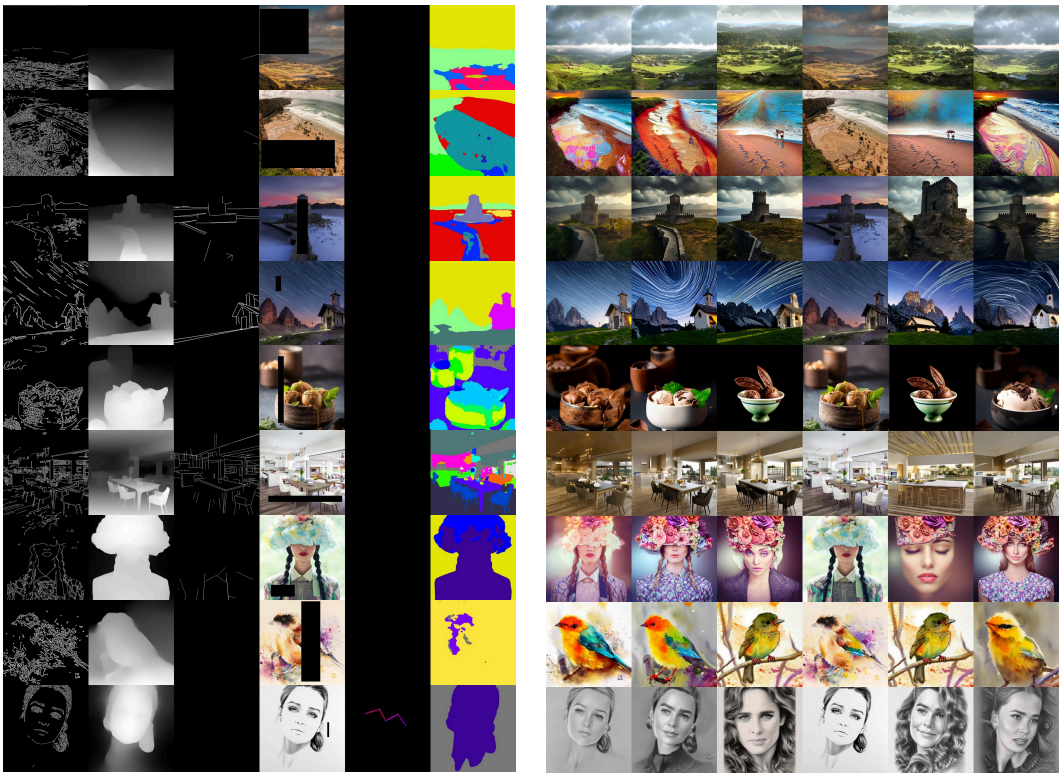


Figure 11: Qualitative results of DeltaControl PixArt-α (20 steps). Zoom in for a better view.

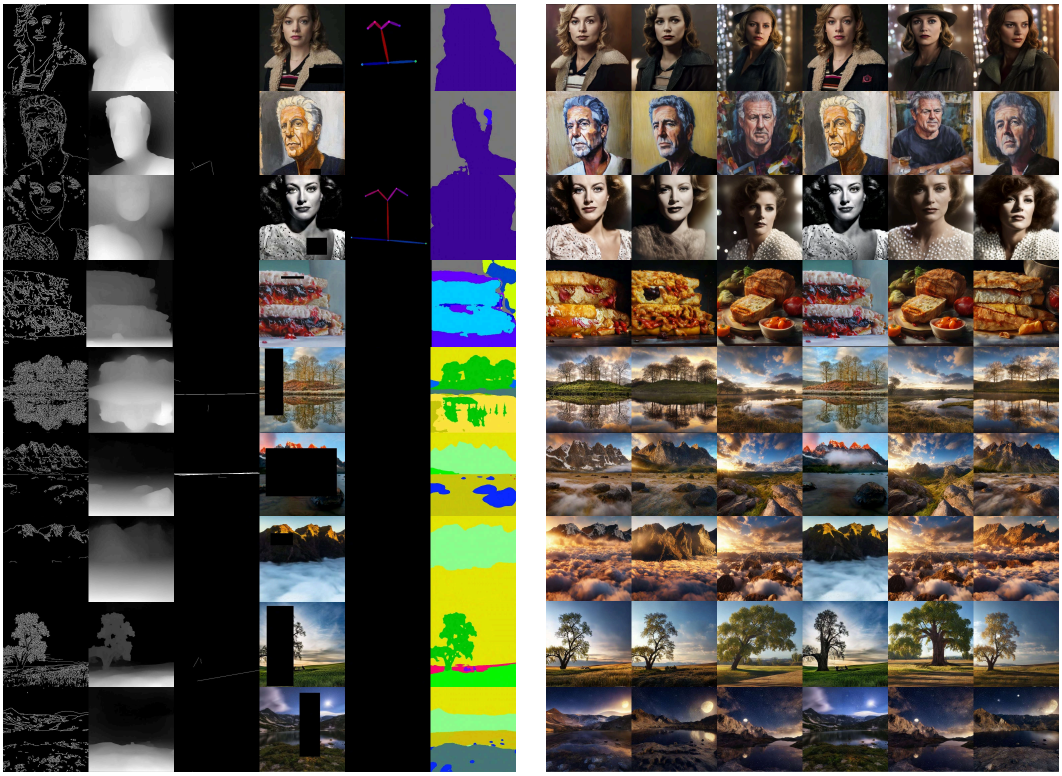


Figure 12: Qualitative results of FlashControl PixArt-α (1 step). Zoom in for a better view.



Figure 13: Qualitative results of DeltaControl FLUX.2-klein-4B (20 steps). Zoom in for a better view.

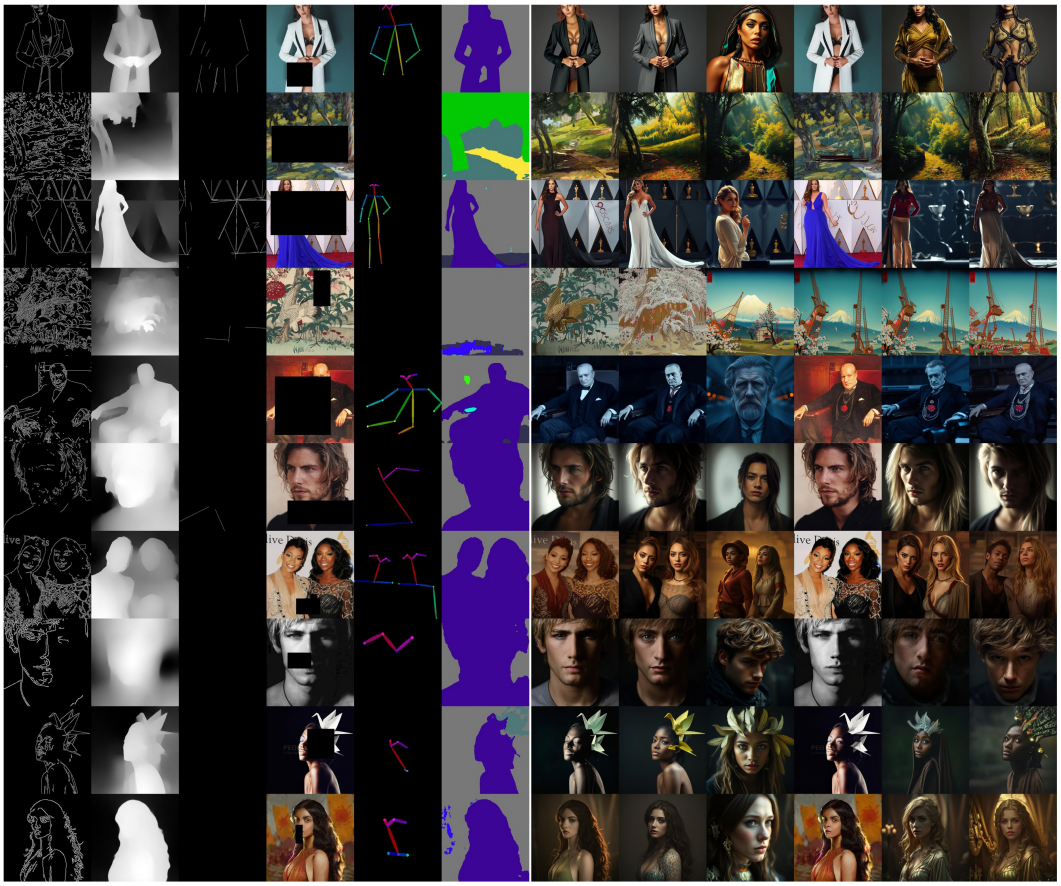


Figure 14: Qualitative results of DeltaControl FLUX.2-klein-4B (20 steps). Zoom in for a better view.

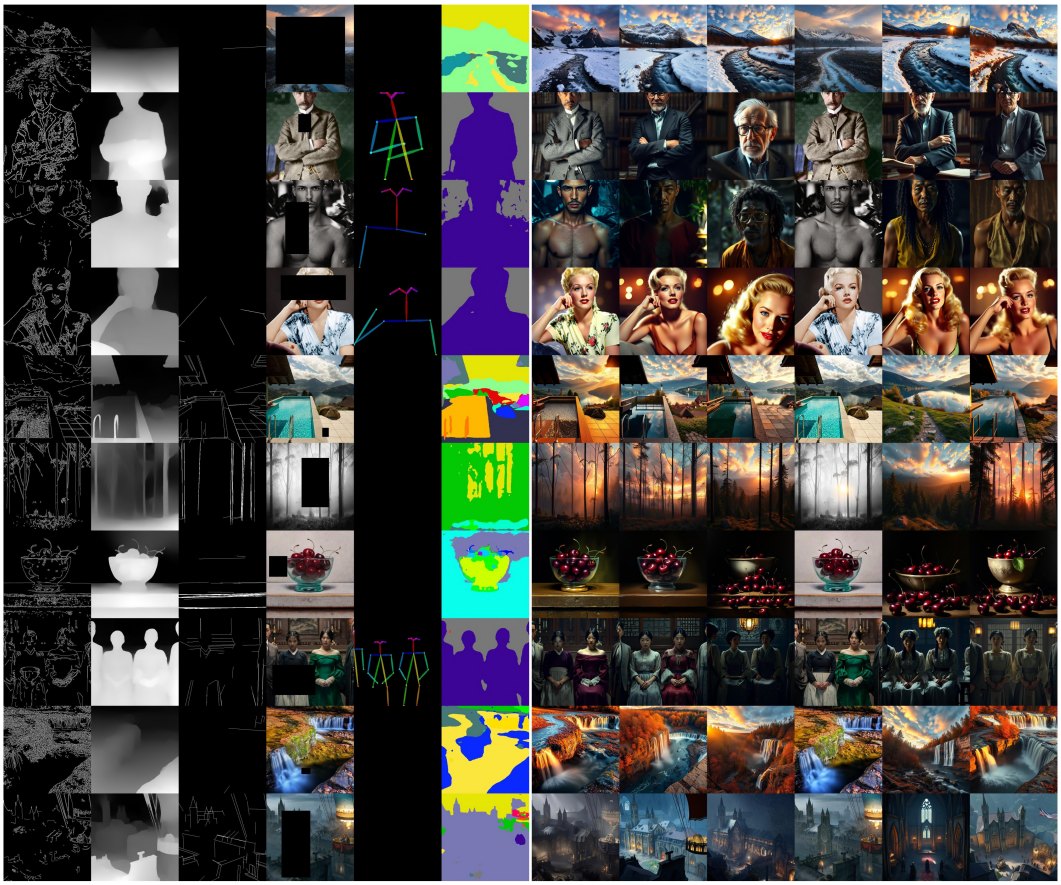


Figure 15: Qualitative results of FlashControl FLUX.2-klein-4B (1 step). Zoom in for a better view.

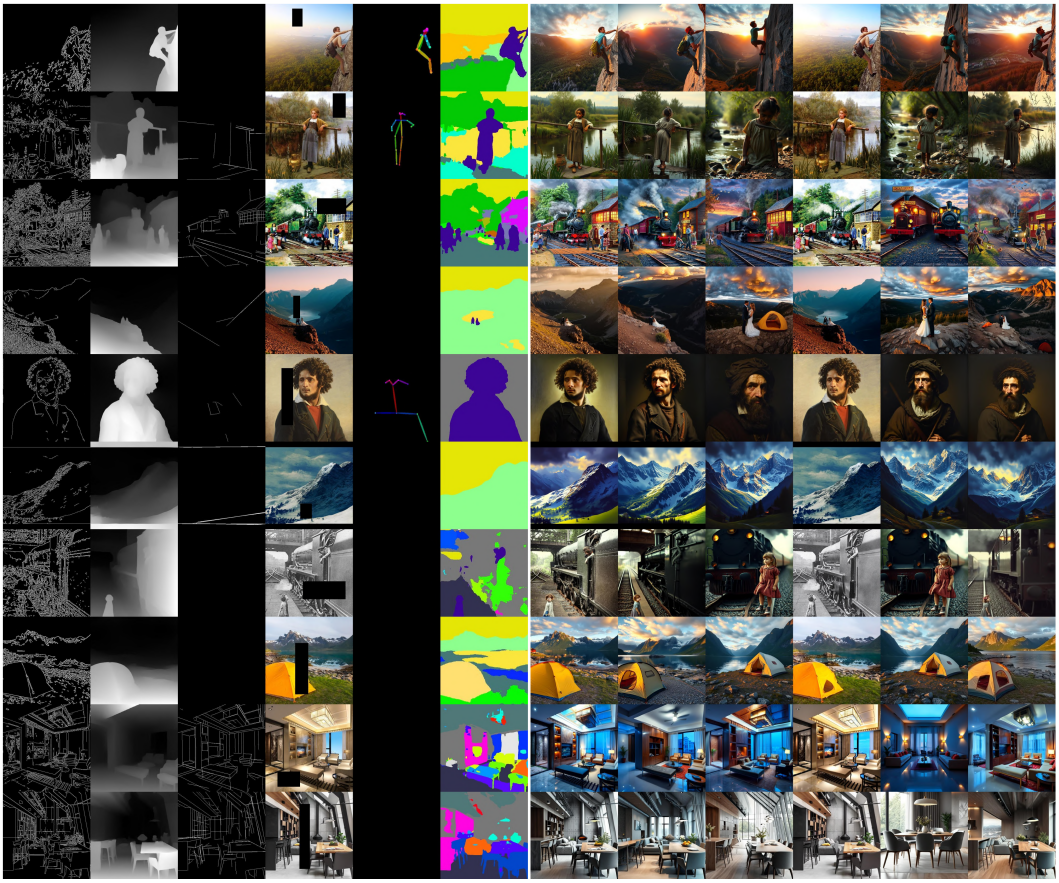


Figure 16: Qualitative results of FlashControl FLUX.2-klein-4B (1 step). Zoom in for a better view.

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: The abstract and introduction accurately state that the paper studies one-step controllable generation using a distillation-friendly single-stream teacher. The claimed contributions are supported by the method, quantitative comparisons, ablations, and efficiency analysis in the main paper and appendix.

   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: The paper includes a limitations section discussing the two-stage training cost, reliance on strong pretrained backbones, evaluation at 512×512, single-control inference, multi-condition scaling, and potential misuse of fast controllable generation.

   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: [N/A]

Justification: The paper does not present formal theoretical results or proofs. The mathematical formulations and first-order arguments are used to motivate the method, while the main claims are empirical.

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: The paper describes the architecture, training stages, objectives, backbones, control modalities, datasets, evaluation protocol, and implementation details needed to reproduce the main experiments.

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: At submission time, we do not provide anonymized code and checkpoints. We plan to release code, configurations, and checkpoints upon publication, subject to the policies of our company.

   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: The paper reports the training data, backbones, control types, LoRA ranks, learning rates, optimizer, resolution, batch size, GPU type, training time, seed, and evaluation protocol in the main paper and appendix.

   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: [No]

   Justification: We do not report error bars or multi-seed confidence intervals because training multiple large diffusion teachers and one-step students is computationally expensive. Instead, we use fixed evaluation sets and matched training protocols for controlled comparisons.

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: The paper reports the GPU type, GPU memory, batch size, resolution, and training time for the main Stage-1 and Stage-2 experiments, as well as training and inference efficiency measurements.

   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: We have reviewed the NeurIPS Code of Ethics and believe the research conforms to it. The submission is anonymized and discusses potential misuse risks of fast controllable 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: The paper discusses positive impacts such as efficient controllable generation, and negative impacts such as lowering the cost of generating misleading visual content. It also discusses responsible release practices such as usage restrictions, provenance tools, and safety filters.

    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: [Yes]

    Justification: The paper discusses safeguards for responsible release, including usage restrictions, provenance or watermarking tools when possible, and content-safety filters inherited from the underlying pretrained generators.

    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: The paper cites the datasets, pretrained models, and baseline methods used in the experiments. We use these assets consistently with their stated licenses and terms of use.

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: The paper does not introduce a new dataset or benchmark. If code or checkpoints are released after publication, they will be documented with training details, intended use, limitations, and license information.

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: [No]

Justification: The paper reports a user study, but the current version does not include the full participant instructions, screenshots, or compensation details. These details should be added to the appendix or supplemental material.

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: [No]

    Justification: The paper includes a user study, but the current version does not state whether IRB approval, exemption, or equivalent review was obtained. This should be clarified in the appendix while preserving anonymity.

    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: The core method, experiments, and scientific claims do not use LLMs as an important, original, or non-standard component. Any writing or formatting assistance does not affect the methodology or results.

    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.