BEYOND ATTENTION MASKS: INSTRUCTION ANCHORING FOR EFFICIENT IN-CONTEXT DIFFUSION GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context diffusion transformers concatenate instruction, target, and reference tokens into a single sequence for joint attention. Reference-side computation must be repeated at every denoising step, with computation growing rapidly as more references are added. Decoupling the reference tokens from the target enables exact K/V reuse across denoising steps. This isolation, however, prevents the reference tokens from attending to the instruction, degrading instruction following and reference fidelity. This trade-off cannot be resolved by attention mask design alone. We introduce **AnchorCache**, a parameter-free token layout and attention mask co-design that inserts static text anchors. The anchors condition the reference representations on the instruction during cache construction, after which the resulting reference K/V are reused exactly across denoising steps. This structural conversion initially degrades generation quality. We recover the lost performance with teacher-forced velocity distillation, followed by a short on-policy stage that queries the teacher at student-visited states. To our knowledge, this is the first use of on-policy distillation for architectural recovery in diffusion models. Across benchmarks spanning image, speech, and video generation, AnchorCache matches full-attention quality. Its efficiency gains increase with the reference context, reaching a $6.40\times$ speedup in DiT inference. Code is available in an anonymous repository.

1 INTRODUCTION

Diffusion transformers increasingly adopt in-context conditioning to integrate heterogeneous inputs such as text, images, audio, and video. In such models, instruction, target, and reference tokens are concatenated into a single sequence for joint attention (Peebles & Xie, 2023; Esser et al., 2024; Zhuang et al., 2025). Although the reference inputs remain unchanged across denoising steps, their layerwise representations depend on the current target through joint attention. This target dependency requires repeating reference-side computation at every denoising step, with the cost growing rapidly as more references are added. At 1024^2 resolution, for example, a single image reference contributes roughly 4k tokens.

A natural way to avoid this repeated computation is to decouple the reference tokens from the target. This makes their layerwise K and V independent of the current denoising step, allowing them to be computed once and reused exactly throughout denoising (Tan et al., 2025b; Zhang et al., 2025c). However, existing cacheable designs enforce this independence by preventing the reference tokens from attending to the instruction states. Without instruction access, the reference representations are less able to capture instruction-relevant content, degrading instruction following and reference fidelity.

The difficulty is that, under full joint attention, the instruction states are themselves contextualized by the evolving target. Allowing reference tokens to attend to these states introduces an indirect dependency on the target, making the reference K/V step-dependent. We therefore need a separate instruction-conditioning path that remains independent of the evolving target.

To this end, we introduce **AnchorCache**. Rather than conditioning the reference branch on the live instruction states, AnchorCache introduces a static copy of the instruction that is isolated from

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

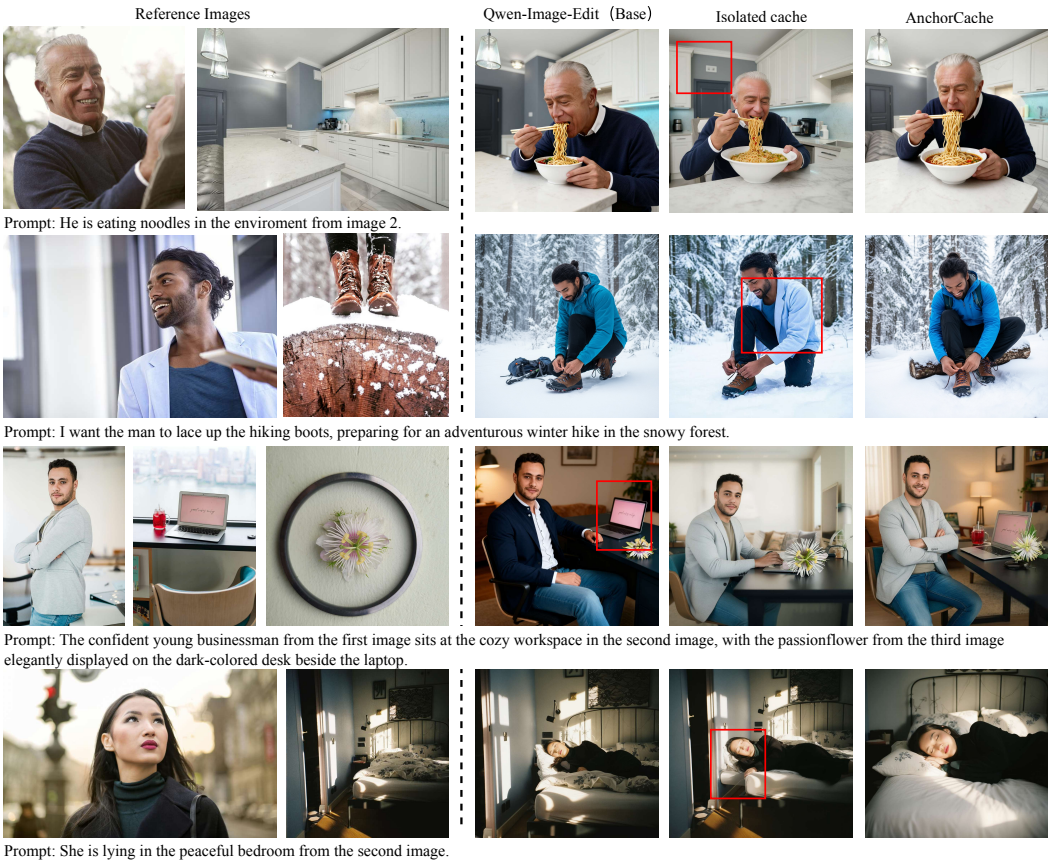

Figure 1: **Qualitative comparison on OmniContext.** Each row shows the reference images and outputs from the full-attention base model, isolated cache, and AnchorCache. Red boxes highlight identity inconsistencies.

the evolving target. This allows the reference branch to incorporate instruction information while preserving exact K/V reuse across denoising steps. The construction introduces no learnable parameters and retains a regular attention layout compatible with optimized implementations such as FlashAttention.

Converting a pretrained full-attention model to this cacheable structure changes its information flow and initially degrades generation quality. We first recover performance through teacher-forced velocity distillation at data-derived states. Since this supervision does not directly cover the states encountered along the student's inference trajectory, we further introduce a short on-policy stage that queries the full-attention teacher at student-visited states. This two-stage procedure targets both the initial quality loss from architectural conversion and residual errors along the student's denoising trajectory.

We evaluate AnchorCache across image, speech, and video generation, including both conventional multi-step and few-step diffusion settings. Across task-specific benchmarks, AnchorCache matches the quality of the corresponding full-attention models.

Our contributions are summarized as follows:

- We introduce **AnchorCache**, a parameter-free token layout and attention mask co-design that makes reference K/V instruction-aware while preserving exact reuse across denoising steps, resolving a structural trade-off that cannot be addressed by attention-mask design alone.
- We develop a two-stage architectural recovery procedure that combines teacher-forced velocity distillation with a short on-policy stage at student-visited states. To our knowledge,

this is the first application of on-policy distillation to architectural recovery in diffusion models.

- We demonstrate AnchorCache across image, speech, and video generation. It matches full-attention quality across task-specific benchmarks, while its efficiency gains increase with the reference context and DiT inference speedups reach $6.40\times$.

2 Related Work

Efficient diffusion transformers. Prior work accelerates diffusion transformers along two broad directions: reducing per-step computation and reusing intermediate states across denoising steps; a complementary line of work reduces the number of sampling steps through distillation (Song et al., 2023). Sparse-attention methods lower per-step cost by evaluating a subset of query–key interactions (Zhang et al., 2025b; Xi et al., 2025; Li et al., 2025b; Wu et al., 2025c; Zhang et al., 2026); however, their dynamic and irregular sparsity complicates kernel implementation, so reductions in attention FLOPs do not always translate into commensurate wall-clock speedups.

Along the reuse direction, temporal caching shares activations across nearby denoising steps (Selvaraju et al., 2024; Liu et al., 2024a; 2025a; Ma et al., 2024; Cui et al., 2026), at the cost of approximation error. Structural methods instead isolate fixed condition tokens from the denoising target, enabling exact K and V reuse (Zhang et al., 2025c; Tan et al., 2025b), but they require additional conditioning modules and leave the cached states unaware of the instruction.

Recovery after architectural changes. Modifying a pretrained diffusion architecture often causes an initial quality drop. Lightweight attention changes can often be recovered through short fine-tuning or distillation (Zhang et al., 2026). More substantial changes replace softmax attention with linear or convolutional operators and typically require longer retraining (Liu et al., 2024b; Dong et al., 2025; Becker et al., 2025; Wang et al., 2025). These methods generally learn from teacher-forced examples drawn from a fixed training distribution.

On-policy distillation. On-policy distillation trains students on states induced by their own trajectories and has been widely studied in language models (Agarwal et al., 2024; Gu et al., 2024; Ko et al., 2025; Yang et al., 2025; Lu & Lab, 2025). Related ideas have also been explored in generative vision for few-step diffusion and autoregressive video generation (Song et al., 2023; Yin et al., 2025; Huang et al., 2025; Cui et al., 2025). These works primarily target sampling acceleration or generative training, rather than recovery from an attention-structure modification.

3 Methodology

3.1 Preliminaries

In-context diffusion transformers. In-context diffusion transformers concatenate instruction, target, and reference tokens into a single sequence for joint attention (Peebles & Xie, 2023; Esser et al., 2024; Tan et al., 2025a; Wu et al., 2025a). Let $\mathrm{LT} \in \mathbb{R}^{N_{\mathrm{LT}} \times d}$ denote the instruction tokens, $X_t \in \mathbb{R}^{N_X \times d}$ the target tokens at denoising step t, and $\mathrm{R}^{(m)} \in \mathbb{R}^{N_{\mathrm{R}}^{(m)} \times d}$ the tokens of the m-th reference input, where d is the hidden dimension. For M reference inputs, we concatenate their tokens as

$$\mathrm{R} = [\mathrm{R}^{(1)}; \ldots; \mathrm{R}^{(M)}], \qquad \mathbf{S}_t = [\mathrm{LT}; X_t; \mathrm{R}], \tag{1}$$

where $N_{\mathrm{R}} = \sum_{m=1}^{M} N_{\mathrm{R}}^{(m)}$ is the total number of reference tokens. The instruction and reference inputs remain fixed throughout denoising, whereas X_t changes at every step. Their layerwise representations, however, may still change across denoising steps through attention to X_t.

Exact reference caching. Let $\mathrm{R}^{(\ell)}(t)$ denote the reference states at transformer layer ℓ and denoising step t. Their attention keys and values are

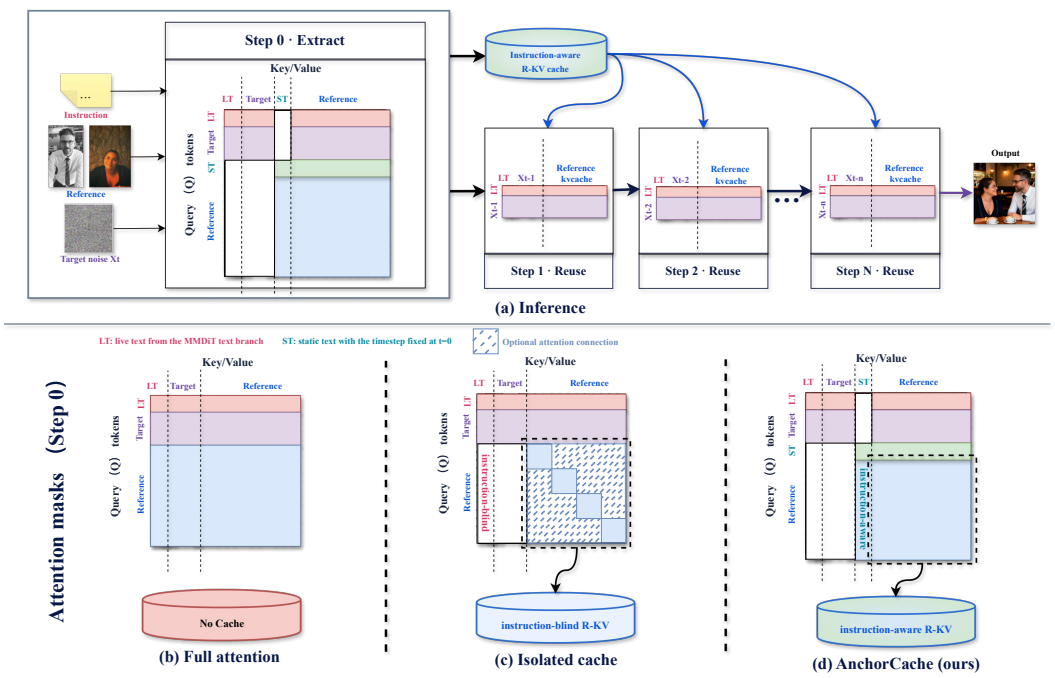

Figure 2: **Overview of AnchorCache.** Top (a): static text anchors condition the reference branch once during cache construction. Only the resulting reference K/V are retained and reused across denoising steps. Bottom: attention structures for (b) full attention, (c) isolated cache, and (d) AnchorCache, where rows denote queries and columns denote keys and values.

$$K_{\mathrm{R}}^{(\ell)}(t) = \mathrm{R}^{(\ell)}(t)W_K^{(\ell)}, \qquad V_{\mathrm{R}}^{(\ell)}(t) = \mathrm{R}^{(\ell)}(t)W_V^{(\ell)}. \tag{2}$$

We call the reference branch *exactly cacheable* when $K_{\mathrm{R}}^{(\ell)}(t)$ and $V_{\mathrm{R}}^{(\ell)}(t)$ remain unchanged across denoising steps at every layer. This property holds by construction when the reference-side inputs are fixed and no step-dependent state, in particular X_t, can reach the reference branch. The reference K/V can then be computed once and reused throughout denoising.

3.2 The Cacheability–Conditioning Trade-off

Exact cacheability does not preserve the conditioning behavior of full attention. As illustrated by the isolated-cache attention mask in Figure 2(c), isolating the reference branch from X_t enables exact K/V reuse, but also removes instruction conditioning from the cached reference representations, weakening instruction–reference binding and potentially degrading reference fidelity (Figure 1).

This limitation becomes more pronounced as the reference context grows. Without instruction-aware reference representations, the dynamic target queries must discriminate instruction-relevant content from an increasingly large set of competing reference tokens. A simple softmax analysis shows that maintaining a fixed attention mass on a relevant token requires the corresponding logit margin to grow logarithmically with the number of competing tokens; we provide the derivation in Appendix A. Thus, when references are instruction-agnostic, instruction-specific retrieval places a greater burden on the dynamic query branch.

Directly allowing references to attend to the original instruction states does not solve this problem: these states are themselves contextualized by X_t, creating the indirect path $X_t \rightarrow \mathrm{LT} \rightarrow \mathrm{R}$ and making the reference states step-dependent. We therefore require reference representations that are instruction-aware yet isolated from the evolving target.

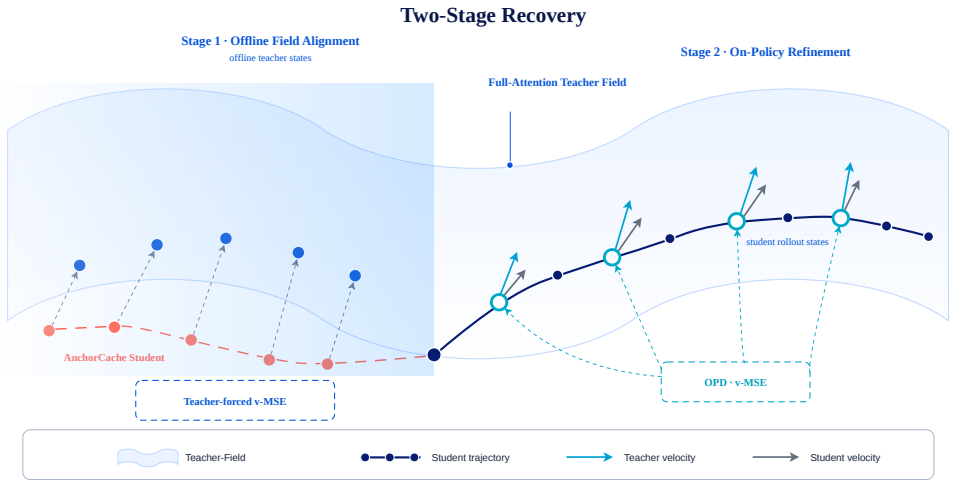

Figure 3: **Two-stage architectural recovery.** Stage 1 queries the teacher at data-derived interpolants. Stage 2 queries the same teacher at states visited by the student and corrects residual errors along the student's inference trajectory.

3.3 AnchorCache: Static Text Anchors

We introduce **AnchorCache**, which adds a static instruction path to condition the reference branch without exposing it to the evolving target.

Static text anchors. We augment $[\mathrm{LT}; X_t; \mathrm{R}]$ with a copy of the instruction tokens, denoted by ST, yielding $[\mathrm{LT}; X_t; \mathrm{ST}; \mathrm{R}]$. The anchors reuse the instruction embeddings with a fixed timestep $t_{\mathrm{ST}} = 0$. As shown in Figure 2(d), ST and R attend to each other, while both are isolated from LT and X_t. The dynamic branch retains the original interaction among LT, X_t, and R, but does not attend to ST. This provides the path $\mathrm{ST} \to \mathrm{R}$ without introducing $X_t \to \mathrm{R}$. The static subgraph $[\mathrm{ST}, \mathrm{R}]$ is computed once before denoising. Only the resulting reference K/V are retained and reused across steps; the anchor states are discarded.

Instruction-aware reference retrieval. Under an isolated cache, target–reference retrieval has the form $q_X(X_t, c)^\top k_R(r)$, where the reference representation itself is instruction-agnostic. AnchorCache instead yields

$$q_X(X_t, c)^\top k_R(r) \quad \longrightarrow \quad q_X(X_t, c)^\top k_R(r, c). \tag{3}$$

Conditional information can therefore be represented on both sides of the attention interaction, rather than primarily through the dynamic target query. This provides additional capacity for resolving the reference competition described in Section 3.2 while preserving exact K/V reuse.

Parameter-free construction. AnchorCache introduces no learnable parameters and reuses the existing instruction embeddings, attention projections, and transformer weights. The computational overhead of introducing static text anchors is analyzed in Appendix D.

3.4 Two-Stage Architectural Recovery

Although AnchorCache enables instruction-aware exact reference caching, directly converting a pretrained full-attention model to the AnchorCache topology initially degrades generation quality because the inherited parameters were optimized under a different attention structure. We recover the lost performance through the two-stage procedure illustrated in Figure 3.

On-policy distillation reduces train–inference mismatch by supervising the student on states induced by its own predictions (Agarwal et al., 2024; Gu et al., 2024). We apply this principle to diffusion

trajectories after an initial teacher-forced recovery stage. Let v_{full} denote the frozen full-attention teacher and v_θ the AnchorCache student.

Stage 1: teacher-forced velocity distillation. Given a clean sample x_0 and Gaussian noise $x_1 \sim \mathcal{N}(0, I)$, we construct the flow-matching interpolant (Lipman et al., 2023)

$$x_t = (1-t)x_0 + tx_1 \tag{4}$$

and minimize

$$\mathcal{L}_{\text{TF}} = \mathbb{E}_{x_0, x_1, c, t}\left[\|v_\theta(x_t, c, t) - v_{\text{full}}(x_t, c, t)\|_2^2\right], \tag{5}$$

where c contains the instruction and reference inputs. The data-derived interpolants determine the query states, while the full-attention model provides the velocity targets. This stage recovers most of the performance lost during architectural conversion. However, it does not expose the student to states reached by its own sampler, so residual errors can remain along the inference trajectory after teacher-forced optimization has largely converged.

Stage 2: on-policy refinement. We roll out the current student from $z_1 \sim \mathcal{N}(0, I)$ and collect the states z_t^θ visited along its denoising trajectory. Let $\bar{z}_t = \text{sg}(z_t^\theta)$ denote the corresponding stop-gradient states. We query the frozen full-attention teacher at these student-visited states and optimize

$$\mathcal{L}_{\text{OPD}} = \mathbb{E}_{z_1, c, t}\left[\|v_\theta(\bar{z}_t, c, t) - v_{\text{full}}(\bar{z}_t, c, t)\|_2^2\right]. \tag{6}$$

The regression target is unchanged from Stage 1; only the query-state distribution changes, from data-derived interpolants to states induced by the student itself. Stop-gradient avoids backpropagation through the sampler. This short on-policy stage directly targets the residual errors encountered along the student's inference trajectory. Appendix B provides further analysis of why changing the query-state distribution can improve recovery once teacher-forced optimization leaves a nonzero residual.

4 Experiments

4.1 Experimental Setup

Models and tasks. We evaluate AnchorCache across four diffusion models spanning image, speech, and video generation: Qwen-Image-Edit-2511 (Wu et al., 2025a), FLUX.2 9B (Black Forest Labs, 2025), OmniVoice (Zhu et al., 2026), and MiniMax-H3 (MiniMax, 2026). We conduct detailed comparisons and ablation studies on Qwen-Image-Edit. For Qwen-Image-Edit, the full-attention baseline uses the officially released weights. The isolated-cache baseline follows the attention structure in Figure 2(c), while AnchorCache uses the structure in Figure 2(d) with static text anchors. Both cacheable variants are initialized from the same full-attention weights. We evaluate AnchorCache on FLUX.2 9B under a 4-step distilled sampling regime, and further extend our evaluation to speech and OmniVideo generation using OmniVoice and MiniMax-H3, respectively.

Evaluation. For image editing, we evaluate on OmniContext (Wu et al., 2025b), GEdit-Bench (Liu et al., 2025b), and ImgEdit-Bench (Ye et al., 2025). OmniContext reports prompt following (PF), subject consistency (SC), and Overall scores; GEdit-Bench reports semantic consistency (SC), perceptual quality (PQ), and Overall scores; and ImgEdit-Bench reports its Overall score. For zero-shot TTS, we evaluate on Seed-TTS EN and Seed-TTS ZH (Anastassiou et al., 2024), and LibriSpeech-PC (Meister et al., 2023), reporting word error rate (WER) and speaker similarity (SIM-o). For OmniVideo generation, we evaluate on OpenVE-Bench (He et al., 2025) using Seed1.6-VL and InternVL3.5-38B as evaluators. We report Overall scores and results for spatially aligned (SA) and non-spatially aligned (NSA) editing categories under each evaluator.

We follow the official evaluation and aggregation protocol of each benchmark; additional training and inference details are provided in Appendix E.

Table 1: Generation quality and inference efficiency across models and modalities. Quality is evaluated on task-specific benchmarks and metrics. Speedup is measured over the complete DiT denoising trajectory relative to the corresponding full-attention model.

Model / Method	Quality							Efficiency
OmniImage Generation	OmniContext			ImgEdit-Bench	GEdit-Bench			Speedup ↑
	PF ↑	SC ↑	Overall ↑	Overall ↑	SC ↑	PQ ↑	Overall ↑	
Qwen-Image-Edit	**8.550**	7.867	8.119	4.324	**8.097**	**7.303**	**7.574**	1.00×
+ Isolated cache	8.155	7.490	7.680	4.219	7.911	7.290	7.475	6.45×
+ **AnchorCache**	8.418	**8.167**	**8.185**	**4.351**	8.059	7.269	7.540	6.40×
FLUX.2 9B *(4 steps)*	8.602	8.043	8.230	3.919	7.650	7.376	7.341	1.00×
+ Isolated cache	8.545	7.527	7.895	3.873	7.331	**7.452**	7.132	2.91×
+ **AnchorCache**	**8.665**	**8.053**	**8.258**	**3.949**	**7.785**	7.431	**7.437**	2.86×
Zero-shot TTS	Seed-TTS EN		Seed-TTS ZH		LibriSpeech-PC			Speedup ↑
	WER (%) ↓	SIM-o ↑	WER (%) ↓	SIM-o ↑	WER (%) ↓	SIM-o ↑		
OmniVoice	**1.62**	**0.7419**	**0.84**	0.7779	1.41	0.7327		1.00×
+ Isolated cache	1.69	0.6985	1.07	0.7687	1.64	0.6960		1.50×
+ **AnchorCache**	1.65	0.7398	0.97	**0.7780**	**1.38**	**0.7327**		1.50×
OmniVideo Generation	OpenVE on Seed1.6-VL			OpenVE on InternVL3.5				Speedup ↑
	Overall ↑	SA ↑	NSA ↑	Overall ↑	SA ↑	NSA ↑		
MiniMax-H3	4.411	4.388	4.525	4.125	4.134	4.082		1.00×
+ Isolated cache	4.398	4.351	4.628	4.153	4.139	**4.224**		1.81×
+ **AnchorCache**	**4.479**	**4.448**	**4.630**	**4.160**	**4.155**	4.187		1.80×

4.2 Main Results

Table 1 summarizes generation quality and inference efficiency across four models spanning image, speech, and video generation. Table 2 then provides a detailed quality–efficiency comparison on Qwen-Image-Edit, including the full-attention model, the isolated-cache baseline, AnchorCache, and representative diffusion acceleration methods.

Results across models and modalities. Table 1 covers substantially different generative settings: Qwen-Image-Edit uses multi-step, multi-reference image editing; FLUX.2 9B uses an aggressively distilled four-step sampler; and we apply the same construction to speech and video with OmniVoice and MiniMax-H3. Across these settings, AnchorCache preserves the quality of the corresponding full-attention model. For example, it obtains an OmniContext Overall score of 8.258 versus 8.230 for FLUX.2 9B, and remains close to the full-attention model on all three speech evaluations. On MiniMax-H3, AnchorCache matches the full-attention teacher's generation quality on OpenVE-Bench, achieving Overall scores of 4.479 versus 4.411 under Seed1.6-VL and 4.160 versus 4.125 under InternVL3.5-38B. These results demonstrate that the same cache construction generalizes across architectures, modalities, and sampling regimes.

Quality and efficiency. AnchorCache substantially reduces end-to-end DiT inference latency while preserving full-attention-level generation quality. Under the setting in Table 2, it achieves a $6.40\times$ speedup, nearly matching the $6.45\times$ isolated cache, which shows that static text anchors add little latency beyond cache construction. At this similar latency, AnchorCache raises the OmniContext Overall score from 7.680 for the isolated cache to 8.185, compared with 8.119 for full attention; the largest gain is in subject consistency, in line with Figure 1. Representative acceleration methods exhibit different quality–efficiency trade-offs and operate on complementary aspects of diffusion inference rather than serving as direct architectural counterparts.

Complementarity with diffusion acceleration methods. AnchorCache is complementary to commonly used diffusion acceleration techniques, such as SageAttention2. Combining the two yields a quality–efficiency trend similar to that observed when applying SageAttention2 to the full-attention baseline, supporting their complementarity. Detailed quantitative results are provided in the appendix C.

Table 2: **Generation quality on OmniContext and inference efficiency.** *All 400* evaluates the complete benchmark. Speedup is measured over the complete 40-step generation process with 8 reference images and is relative to the full-attention base model.

	MULTIPLE		SCENE		*All 400*			Efficiency
Method	PF ↑	SC ↑	PF ↑	SC ↑	PF ↑	SC ↑	Overall ↑	Speedup ↑
Base (Qwen-Image-Edit)	<u>8.660</u>	8.020	**8.180**	<u>7.693</u>	**8.550**	7.867	<u>8.119</u>	1.00×
SVDQuant (Li et al., 2025a)	7.913	7.807	7.247	6.967	7.888	7.520	7.582	1.49×
TeaCache (Liu et al., 2025a)	8.513	7.873	7.813	7.427	8.330	7.675	7.896	3.56×
SageAttention2 (Zhang et al., 2025a)	**8.720**	<u>8.280</u>	<u>7.993</u>	7.580	<u>8.495</u>	<u>7.923</u>	8.105	1.47×
ToPi (Lin et al., 2026)	8.580	8.107	7.940	7.447	8.400	7.692	7.891	1.42×
Isolated cache	8.053	7.967	7.720	7.287	8.155	7.490	7.680	6.45×
AnchorCache (ours)	8.573	**8.453**	7.913	**7.947**	8.418	**8.167**	**8.185**	6.40×

Table 3: **Ablation of static text anchors and recovery stages.** Results are reported across three image-editing benchmarks for the full-attention model and both cacheable architectures after Stage 1 and Stage 2 recovery.

	OmniContext			ImgEdit-Bench	GEdit-Bench		
Method	PF ↑	SC ↑	Overall ↑	Overall ↑	SC ↑	PQ ↑	Overall ↑
Base (Qwen-Image-Edit)	**8.550**	7.867	<u>8.119</u>	<u>4.324</u>	**8.097**	**7.303**	**7.574**
Isolated, Stage 1	8.180	7.203	7.529	4.184	7.929	7.181	7.440
Isolated, + Stage 2	8.155	7.490	7.680	4.219	7.911	<u>7.290</u>	7.475
AnchorCache, Stage 1	8.285	<u>7.970</u>	8.019	4.293	7.940	7.252	7.460
AnchorCache, + Stage 2	<u>8.418</u>	**8.167**	**8.185**	**4.351**	<u>8.059</u>	7.269	<u>7.540</u>

4.3 ABLATION STUDY

Effect of static text anchors and recovery stages. Table 3 separates the effects of the attention topology and the two recovery stages. At the same recovery stage, AnchorCache consistently outperforms the isolated cache across all three benchmarks. On OmniContext, for example, static anchors improve Stage 1 Overall from 7.529 to 8.019; after Stage 2, AnchorCache reaches 8.185 versus 7.680 for the isolated cache. Stage 2 also improves AnchorCache on ImgEdit-Bench and GEdit-Bench, showing that instruction-aware static anchors and on-policy refinement provide complementary gains.

Teacher-forced saturation and on-policy recovery. We next test whether the Stage 2 gain can be obtained by simply continuing teacher-forced distillation. Starting from the same 30k Stage 1 checkpoint, continued teacher-forced training remains nearly saturated across the 30k–33k checkpoints, with an Overall mean of 8.015 ± 0.015. In contrast, on-policy refinement reaches 8.057 after only 250 updates and 8.185 after 500, with every evaluated on-policy checkpoint exceeding the best continued teacher-forced checkpoint (Figure 4). The remaining error is therefore better addressed by supervising the model at student-visited states than by further optimization on data-derived interpolants.

OPD query design. We vary both the query interval and the number of states sampled from each student rollout. Increasing the number of sampled states does not uniformly improve recovery: its benefit depends strongly on where the queries are drawn from along the trajectory. In particular, concentrating queries in a restricted noise range can even degrade performance, whereas sampling across the full trajectory consistently performs best. We therefore adopt full-trajectory sampling with $N_q = 4$ as our default OPD configuration; further experimental details are provided in Appendix E.1.

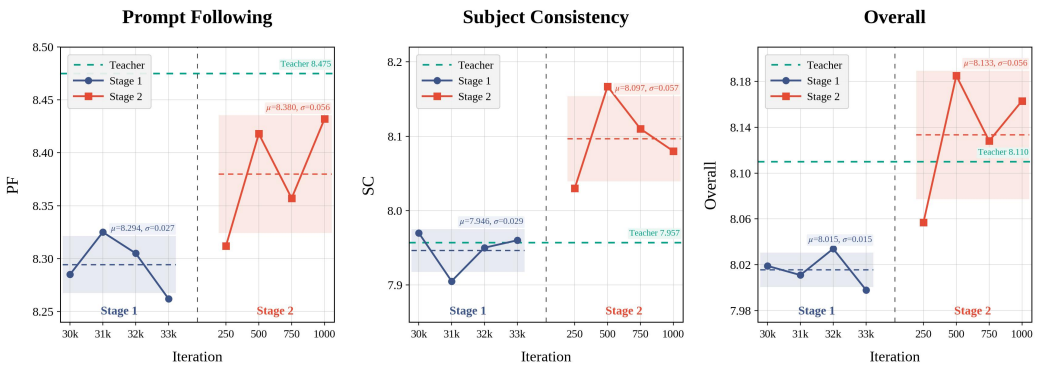

Figure 4: **Teacher-forced saturation and on-policy recovery.** Starting from the same 30k Stage 1 checkpoint, we compare continued teacher-forced training with Stage 2 on-policy recovery. Shaded regions denote mean $\pm$ standard deviation, and the green dashed line indicates the full-attention teacher.

Table 4: **OPD query design.** All configurations are initialized from the same 30k Stage 1 checkpoint and evaluated after 500 on-policy updates. Overall scores are reported for different query intervals and numbers of sampled states N_q, with Δ measured from the Stage 1 starting point (8.019).

	$N_q = 1$		$N_q = 4$	
Query interval	Overall	Δ	Overall	Δ
Full trajectory	8.084	+0.065	**8.185**	**+0.166**
High-noise	8.060	+0.041	8.131	+0.112
Mid-noise	8.073	+0.054	8.051	+0.032
Low-noise	7.936	−0.083	7.888	−0.131

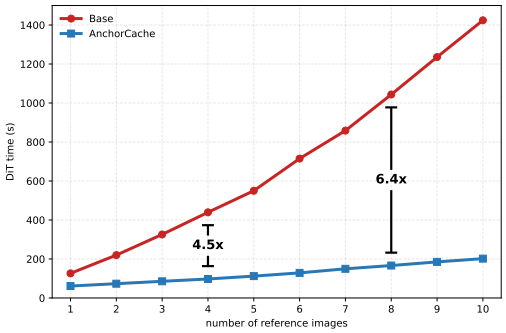

Figure 5: **Scaling with the number of references.** DiT inference latency as the number of reference images increases.

Scaling with the number of references. We further evaluate how DiT denoising latency scales with the number of reference images using Qwen-Image-Edit (20B). Under full attention, every additional reference introduces tokens that must be repeatedly processed throughout the denoising trajectory, causing latency to grow rapidly with the reference context. In contrast, AnchorCache computes the reference representations once and reuses the cached K/V at subsequent denoising steps. Consequently, its relative efficiency gain increases with the number of references, as shown in Figure 5.

5 Conclusion

In this work, we studied efficient inference for in-context diffusion models. Exact reference caching requires the reference representations to remain independent of the denoising target, but isolating the reference branch also removes instruction conditioning. We introduced **AnchorCache**, a parameter-free token layout and attention mask co-design that uses static text anchors to condition the reference representations during cache construction while preserving exact K/V reuse. To restore generation quality after this architectural conversion, we developed a two-stage procedure combining teacher-forced velocity distillation with short on-policy refinement at student-visited states. Across image, speech, and video generation, AnchorCache preserves the quality of the corresponding full-attention models while avoiding repeated reference-side computation across denoising steps. Its efficiency advantage further increases as the reference context grows. Overall, our results show that exact reference caching need not come at the cost of instruction-aware conditioning, providing a practical approach to more efficient diffusion generation.

AI USE STATEMENT

Generative AI tools were used to improve the clarity and readability of the manuscript, assist with manuscript organization and formatting, and provide feedback during brainstorming about the methodology, experimental design, and interpretation of results. The authors independently evaluated all suggestions, verified all technical claims and reported results, and reviewed all AI-assisted content. The authors take full responsibility for the final content of this work.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

Philip Anastassiou, Jiawei Chen, Jitong Chen, Yuanzhe Chen, Zhuo Chen, Ziyi Chen, Jian Cong, Lelai Deng, Chuang Ding, Lu Gao, et al. Seed-TTS: A family of high-quality versatile speech generation models, 2024.

Philipp Becker, Abhinav Mehrotra, Ruchika Chavhan, Malcolm Chadwick, Luca Morreale, Mehdi Noroozi, Alberto Gil Ramos, and Sourav Bhattacharya. EDiT: Efficient diffusion transformers with linear compressed attention, 2025.

Black Forest Labs. FLUX.2: Frontier visual intelligence. Black Forest Labs technical blog, 2025. https://bfl.ai/blog/flux-2.

Guoxuan Chen, Chufeng Xiao, Haoran Yang, Siyue Xie, Binxiao Huang, Ming Zhang, Cheuk Him Chau, Xinyu Fu, Yingzhao Lian, Tom S. Y. Li, et al. Boogu-Image-0.1: Boosting open-source unified multimodal understanding and generation, 2026.

Hanshuai Cui, Zhiqing Tang, Zhifei Xu, Zhi Yao, Wenyi Zeng, and Weijia Jia. BWCache: Accelerating video diffusion transformers through block-wise caching, 2026.

Justin Cui, Jie Wu, Ming Li, Tao Yang, Xiaojie Li, Rui Wang, Andrew Bai, Yuanhao Ban, and Cho-Jui Hsieh. Self-Forcing++: Towards minute-scale high-quality video generation, 2025.

ZiYi Dong, Chengxing Zhou, Weijian Deng, Pengxu Wei, Xiangyang Ji, and Liang Lin. Can we achieve efficient diffusion without self-attention? distilling self-attention into convolutions, 2025.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-policy distillation of large language models. In *International Conference on Learning Representations*, 2024.

Haorui He, Zengqiang Shang, Chaoren Wang, Xuyuan Li, Yicheng Gu, Hua Hua, Liwei Liu, Chen Yang, Jiaqi Li, Peiyang Shi, Yuancheng Wang, Kai Chen, Pengyuan Zhang, and Zhizheng Wu. Emilia: An extensive, multilingual, and diverse speech dataset for large-scale speech generation, 2024.

Haoyang He, Jie Wang, Jiangning Zhang, Zhucun Xue, Xingyuan Bu, Qiangpeng Yang, Shilei Wen, and Lei Xie. OpenVE-3M: A large-scale high-quality dataset for instruction-guided video editing, 2025.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, 2025.

Zeyu Jiang, Lai Man Po, Xuyuan Xu, Yexin Wang, Guoping Gong, Haoxuan Wu, Chenbo Yan, Kun Li, and Yuyang Liu. OrionEdit: Bridging reference and source images for generalized cross-image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9127–9138, 2026.

Jongwoo Ko, Tianyi Chen, Sungnyun Kim, Tianyu Ding, Luming Liang, Ilya Zharkov, and Se-Young Yun. DistiLLM-2: A contrastive approach boosts the distillation of LLMs. In *International Conference on Machine Learning*, 2025.

Bingnan Li, Haozhe Wang, Haozhong Xiong, Fangtai Wu, Jinpeng Yu, Yang Shi, Jiaming Liu, and Ruihua Huang. Rethinking classifier-free guidance in on-policy diffusion distillation, 2026.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. SVDQuant: Absorbing outliers by low-rank components for 4-bit diffusion models. In *International Conference on Learning Representations*, 2025a.

Xingyang Li, Muyang Li, Tianle Cai, Haocheng Xi, Shuo Yang, Yujun Lin, Lvmin Zhang, Songlin Yang, Jinbo Hu, Kelly Peng, Maneesh Agrawala, Ion Stoica, Kurt Keutzer, and Song Han. Radial attention: $O(n \log n)$ sparse attention with energy decay for long video generation. In *Advances in Neural Information Processing Systems*, 2025b.

Junqing Lin, Xingyu Zheng, Pei Cheng, Bin Fu, Jingwei Sun, and Guangzhong Sun. Token pruning for in-context generation in diffusion transformers, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Feng Liu, Shiwei Zhang, Xiaofeng Wang, Yujie Wei, Haonan Qiu, Yuzhong Zhao, Yingya Zhang, Qixiang Ye, and Fang Wan. Timestep embedding tells: It's time to cache for video diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7353–7363, 2025a.

Joseph Liu, Joshua Geddes, Ziyu Guo, Haomiao Jiang, and Mahesh Kumar Nandwana. SmoothCache: A universal inference acceleration technique for diffusion transformers, 2024a.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, Guopeng Li, Yuang Peng, Quan Sun, Jingwei Wu, Yan Cai, Zheng Ge, Ranchen Ming, Lei Xia, Xianfang Zeng, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, Gang Yu, and Daxin Jiang. Step1X-Edit: A practical framework for general image editing, 2025b.

Songhua Liu, Zhenxiong Tan, and Xinchao Wang. CLEAR: Conv-like linearization revs pre-trained diffusion transformers up, 2024b.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation/.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. In *Advances in Neural Information Processing Systems*, 2024.

Aleksandr Meister, Matvei Novikov, Nikolay Karpov, Evelina Bakhturina, Vitaly Lavrukhin, and Boris Ginsburg. LibriSpeech-PC: Benchmark for evaluation of punctuation and capitalization capabilities of end-to-end ASR models, 2023.

Meituan LongCat Team, Hanghang Ma, Haoxian Tan, Jiale Huang, Junqiang Wu, Jun-Yan He, Lishuai Gao, Songlin Xiao, Xiaoming Wei, Xiaoqi Ma, Xunliang Cai, Yayong Guan, and Jie Hu. LongCat-Image technical report, 2025.

MiniMax. MiniMax H3: An open model breaking the boundaries between tasks and modalities. MiniMax Research Blog, 2026. Released July 31, 2026.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Pratheba Selvaraju, Tianyu Ding, Tianyi Chen, Ilya Zharkov, and Luming Liang. FORA: Fast-forward caching in diffusion transformer acceleration, 2024.

Lin Song, Wenbo Li, Guoqing Ma, Wei Tang, Bo Wang, Yuan Zhang, Yijun Yang, Yicheng Xiao, Jianhui Liu, Yanbing Zhang, et al. Awaking spatial intelligence in unified multimodal understanding and generation, 2026.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

Super Intelligence Team, Changhao Qiao, Chao Hui, Chen Li, Cunzheng Wang, Dejia Song, Jiale Zhang, Jing Li, Qiang Xiang, Runqi Wang, Shuang Sun, Wei Zhu, Xu Tang, Yao Hu, Yibo Chen, Yuhao Huang, Yuxuan Duan, Zhiyi Chen, and Ziyuan Guo. FireRed-Image-Edit-1.0 technical report, 2026.

Zhenxiong Tan, Songhua Liu, Xingyi Yang, Qiaochu Xue, and Xinchao Wang. OminiControl: Minimal and universal control for diffusion transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025a.

Zhenxiong Tan, Qiaochu Xue, Xingyi Yang, Songhua Liu, and Xinchao Wang. OminiControl2: Efficient conditioning for diffusion transformers, 2025b.

Jiahao Wang, Ning Kang, Lewei Yao, Mengzhao Chen, Chengyue Wu, Songyang Zhang, Shuchen Xue, Yong Liu, Taiqiang Wu, Xihui Liu, Kaipeng Zhang, Shifeng Zhang, Wenqi Shao, Zhenguo Li, and Ping Luo. LiT: Delving into a simple linear diffusion transformer for image generation, 2025.

Cong Wei, Zheyang Xiong, Weiming Ren, Xinrun Du, Ge Zhang, and Wenhu Chen. OmniEdit: Building image editing generalist models through specialist supervision. In *International Conference on Learning Representations*, 2025.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-Image technical report, 2025a.

Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, Ze Liu, Ziyi Xia, Chaofan Li, Haoge Deng, Jiahao Wang, Kun Luo, Bo Zhang, Defu Lian, Xinlong Wang, Zhongyuan Wang, Tiejun Huang, and Zheng Liu. Omnigen2: Exploration to advanced multimodal generation. *arXiv preprint arXiv:2506.18871*, 2025b.

Jianzong Wu, Liang Hou, Haotian Yang, Xin Tao, Ye Tian, Pengfei Wan, Di Zhang, and Yunhai Tong. VMoBA: Mixture-of-block attention for video diffusion models, 2025c.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, Jianfei Chen, Ion Stoica, Kurt Keutzer, and Song Han. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity, 2025.

An Yang et al. Qwen3 technical report, 2025.

Yang Ye, Xianyi He, Zongjian Li, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, and Li Yuan. ImgEdit: A unified image editing dataset and benchmark, 2025.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jintao Zhang, Haofeng Huang, Pengle Zhang, Jia Wei, Jun Zhu, and Jianfei Chen. SageAttention2: Efficient attention with thorough outlier smoothing and per-thread INT4 quantization. In *International Conference on Machine Learning*, pp. 75097–75119, 2025a.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *International Conference on Machine Learning*, 2025b.

Jintao Zhang, Kai Jiang, Chendong Xiang, Weiqi Feng, Yuezhou Hu, Haocheng Xi, Jianfei Chen, and Jun Zhu. SpargeAttention2: Trainable sparse attention via hybrid top-k+top-p masking and distillation fine-tuning, 2026.

Yuxuan Zhang, Yirui Yuan, Yiren Song, Haofan Wang, and Jiaming Liu. EasyControl: Adding efficient and flexible control for diffusion transformer, 2025c.

Han Zhu, Lingxuan Ye, Wei Kang, Zengwei Yao, Liyong Guo, Fangjun Kuang, Zhifeng Han, Weiji Zhuang, Long Lin, and Daniel Povey. OmniVoice: Towards omnilingual zero-shot text-to-speech with diffusion language models, 2026.

Junhao Zhuang, Lingen Li, Xuan Ju, Zhaoyang Zhang, Chun Yuan, and Ying Shan. Cobra: Efficient line art COlorization with BRoAder references, 2025.

A Reference Competition and Query-Side Discrimination

This section gives the general form of the reference-competition argument used in Section 3.2. Consider a target query q, one instruction-relevant reference key k_+, and N competing reference keys $\{k_i\}_{i=1}^N$. Let their attention logits be

$$a = \frac{q^\top k_+}{\sqrt{d}}, \qquad b_i = \frac{q^\top k_i}{\sqrt{d}}. \tag{7}$$

The attention probability assigned to the relevant token is

$$p_+ = \frac{e^a}{e^a + \sum_{i=1}^N e^{b_i}}. \tag{8}$$

Requiring $p_+ \geq \rho$ is equivalent to

$$a - \log \sum_{i=1}^N e^{b_i} \geq \log \frac{\rho}{1-\rho}. \tag{9}$$

Define the effective competing logit as

$$b_{\text{eff}} = \log \left(\frac{1}{N} \sum_{i=1}^N e^{b_i} \right). \tag{10}$$

Since $\log \sum_i e^{b_i} = \log N + b_{\text{eff}}$, Eq. equation 9 becomes

$$a - b_{\text{eff}} \geq \log N + \log \frac{\rho}{1-\rho}. \tag{11}$$

Thus, for a fixed desired attention mass ρ, the required separation between the relevant token and the effective competing reference context grows as $\log N$. When all competing logits are equal, $b_i = b$, the general expression reduces to the equal-logit case.

The equal-logit case is therefore not required by the argument; it only gives a convenient closed-form illustration. More generally, increasing the reference context increases the amount of competing attention mass that must be separated from instruction-relevant content.

Under an isolated cache, the reference keys $k_R(r)$ are independent of the instruction, so instruction-specific discrimination must be expressed primarily through the dynamic target query $q_X(X_t, c)$. AnchorCache instead constructs instruction-conditioned reference keys $k_R(r, c)$, allowing the conditioning signal to shape both sides of the retrieval interaction. This does not remove reference competition, but provides additional representational capacity for resolving it while preserving exact K/V reuse.

B Why the Recovery Is Staged

Stage 1: establishing a useful student trajectory. Converting a pretrained full-attention model to AnchorCache changes its information flow and initially degrades generation quality. Stage 1 uses data-derived interpolants to recover the student's prediction behavior before relying on its own rollouts. Beyond improving generation quality, this provides a useful initialization for Stage 2: the student can generate reasonably coherent intermediate states at which the teacher is queried.

This initialization is important because on-policy distillation queries the teacher at student-visited states rather than replacing the preceding student trajectory with a teacher-generated one. If the student has already accumulated large errors, its states may deviate substantially from those encountered during successful generation. Although the teacher can still predict a velocity at these states,

matching that velocity locally does not guarantee correction of errors accumulated earlier in the trajectory. Stage 1 therefore helps establish a useful rollout distribution before on-policy refinement.

Stage 2: refining student-visited states. Even after teacher-forced recovery, the student may encounter states that differ from the data-derived interpolants used during Stage 1. Stage 2 addresses this train–inference mismatch by querying the full-attention teacher at states visited along the student's own denoising trajectory. The teacher and student are evaluated at the same detached states, so the student receives supervision on its actual inference distribution without backpropagating through the rollout. As the student is updated, subsequent rollouts expose the teacher to the student's evolving states.

Motivation for the training order. The two stages thus serve complementary purposes: Stage 1 recovers a useful initial generation trajectory, while Stage 2 targets residual errors along student-visited states. Empirically, continued teacher-forced training plateaus, whereas switching to on-policy refinement yields further improvements (Figure 4). This motivates our staged schedule without implying that teacher-forced initialization is mathematically necessary for on-policy distillation.

C Compatibility with Common Diffusion Acceleration Techniques

We evaluate whether AnchorCache can be combined with commonly used diffusion acceleration techniques. Specifically, we apply SageAttention2 (Zhang et al., 2025a) to AnchorCache and evaluate the combined method on Qwen-Image-Edit (20B) using OmniContext. AnchorCache removes repeated reference-side computation across denoising steps, while remaining compatible with additional inference acceleration. This experiment assesses their practical complementarity.

Evaluation. Table 5 compares the full-attention baseline, the baseline with SageAttention2, AnchorCache, and AnchorCache with SageAttention2. We report prompt following (PF), subject consistency (SC), and Overall scores, following the evaluation protocol described in Appendix E. The first three rows reproduce the corresponding results from Table 2. All speedups are relative to the same full-attention baseline.

Table 5: **Combining AnchorCache with SageAttention2.** Quality is evaluated on OmniContext, with *All 400* covering the complete benchmark. Speedup is measured over the complete 40-step DiT denoising process with 8 reference images, relative to the full-attention baseline.

	Multiple		Scene		*All 400*			Efficiency
Method	PF ↑	SC ↑	PF ↑	SC ↑	PF ↑	SC ↑	Overall ↑	Speedup ↑
Full-attention baseline	8.660	8.020	8.180	7.693	8.550	7.867	8.119	1.00×
+ SageAttention2	8.720	8.280	7.993	7.580	8.495	7.923	8.105	1.47×
AnchorCache	8.573	8.453	7.913	7.947	8.418	8.167	**8.185**	6.40×
+ SageAttention2	8.41	8.32	7.69	7.59	8.31	8.01	8.062	**7.99**×

Additional acceleration. Combining AnchorCache with SageAttention2 increases the speedup from $6.40\times$ to $7.99\times$ relative to the full-attention baseline. This corresponds to an additional $1.25\times$ speedup over AnchorCache alone, or approximately 19.9% lower latency. The combined method has a measured latency of 130.4 seconds under this setting.

Quality–efficiency trade-off. The additional acceleration is accompanied by a decrease in the Overall score from 8.185 to 8.062. Applying SageAttention2 to the full-attention baseline similarly improves efficiency while reducing the Overall score, from 8.119 to 8.105. The two settings therefore exhibit the same qualitative direction of the trade-off, although the quality reduction is

larger when SageAttention2 is combined with AnchorCache (0.123 versus 0.014 points). These results demonstrate that AnchorCache and SageAttention2 can be combined to obtain an additional quality–efficiency operating point, supporting their practical complementarity.

D Compute Scaling with Reference Context

We analyze how the computation of full attention, isolated caching, and AnchorCache scales with the number of reference tokens. Let N_{R} denote the number of reference tokens, N_{ST} the number of static-text tokens, and

$$S = N_X + N_{\mathrm{LT}} \tag{12}$$

the number of live target and instruction tokens. We omit constant factors and focus on the dominant token-dependent terms in a transformer block.

Full attention. Under full attention, all $N_{\mathrm{R}} + S$ tokens are updated at every denoising step. Over T denoising steps, the computation therefore scales as

$$\mathcal{C}_{\mathrm{full}} = O\big(T(N_{\mathrm{R}} + S)d^2 + T(N_{\mathrm{R}} + S)^2 d\big)\,, \tag{13}$$

where the first term represents token-wise projections and feed-forward computation, and the second represents attention.

Isolated cache. Once the reference branch is isolated from the live target and instruction states, its representations can be computed once and reused across all denoising steps. Cache construction over the reference tokens costs

$$\mathcal{C}_{\mathrm{iso,pre}} = O\big(N_{\mathrm{R}} d^2 + N_{\mathrm{R}}^2 d\big)\,. \tag{14}$$

After cache construction, only the S live tokens are updated. Their queries still attend to the cached reference K/V, giving a repeated cost

$$\mathcal{C}_{\mathrm{live}} = O\big(TSd^2 + TS(N_{\mathrm{R}} + S)d\big)\,. \tag{15}$$

The total isolated-cache cost is therefore

$$\mathcal{C}_{\mathrm{iso}} = O\big(N_{\mathrm{R}} d^2 + N_{\mathrm{R}}^2 d + TSd^2 + TS(N_{\mathrm{R}} + S)d\big)\,. \tag{16}$$

AnchorCache. AnchorCache has the same live computation as the isolated cache. The only additional computation comes from including the static instruction tokens ST in the one-time cache-construction subgraph $[\mathrm{ST}, \mathrm{R}]$. Its total cost is

$$\mathcal{C}_{\mathrm{anchor}} = O\big((N_{\mathrm{R}} + N_{\mathrm{ST}})d^2 + (N_{\mathrm{R}} + N_{\mathrm{ST}})^2 d + TSd^2 + TS(N_{\mathrm{R}} + S)d\big)\,. \tag{17}$$

Importantly, Eqs. equation 16 and equation 17 have identical repeated per-step computation. AnchorCache therefore introduces no additional reference-dependent cost during the denoising trajectory.

The extra computation relative to the isolated cache is confined to cache construction:

$$\Delta\mathcal{C}_{\mathrm{anchor}} = \mathcal{C}_{\mathrm{anchor}} - \mathcal{C}_{\mathrm{iso}} \tag{18}$$

$$= O\big(N_{\mathrm{ST}} d^2 + \big(2N_{\mathrm{R}} N_{\mathrm{ST}} + N_{\mathrm{ST}}^2\big)\, d\big)\,. \tag{19}$$

Since the static-text block is computed only once, this overhead is amortized over the full denoising trajectory. Moreover, $N_{\mathrm{ST}} = N_{\mathrm{LT}}$ in AnchorCache and is independent of the number of references. Consequently, as the reference context grows, the dominant saving remains the removal of repeated reference-side computation, while the additional cost of instruction-aware cache construction remains a one-time overhead.

Scaling with reference context. The key difference is the reference-dependent repeated computation. Full attention repeatedly incurs quadratic reference-side attention,

$$O(TN_{\mathrm{R}}^2), \tag{20}$$

whereas both cacheable variants pay the quadratic reference cost only once and reduce the repeated reference-dependent term to

$$O(TSN_{\mathrm{R}}). \tag{21}$$

AnchorCache therefore preserves the same asymptotic denoising-time complexity as the isolated cache while adding only the one-time overhead in Eq. equation 19. This is consistent with the measured results: AnchorCache achieves nearly the same inference speedup as the isolated cache while substantially recovering generation quality.

This analysis describes structural compute scaling rather than exact wall-clock latency. Actual latency additionally depends on memory traffic, kernel-launch overhead, hardware utilization, and other model components.

E Experimental Details

E.1 OPD Query Sampling

We vary both the region of the denoising trajectory from which supervision states are drawn and the number of query states N_q collected from each student rollout. All configurations are initialized from the same 30k Stage 1 checkpoint, whose OmniContext Overall score is 8.019.

For the full-trajectory setting, query states are sampled across the complete student denoising trajectory. The restricted settings use only the corresponding portions of the 40-step rollout:

$$\text{High-noise:} \quad k \in [1, 14], \tag{22}$$

$$\text{Mid-noise:} \quad k \in [15, 24], \tag{23}$$

$$\text{Low-noise:} \quad k \in [25, 39], \tag{24}$$

$$\text{Full trajectory:} \quad k \in [1, 39]. \tag{25}$$

Here k denotes the rollout step index. For each configuration, the N_q query states are sampled only from the specified interval.

The results in Table 4 show that increasing N_q alone does not guarantee better recovery. In particular, additional queries concentrated in the low-noise region degrade performance, whereas sampling across the full trajectory performs best for both evaluated values of N_q. We therefore use full-trajectory sampling with $N_q = 4$ as the default Stage 2 configuration.

Recovery hyperparameters. Table 6 summarizes the settings used for the two recovery stages. We use a relatively tight gradient-norm clip of 0.05, which we found empirically stable for the small recovery updates in this setting. Stage 2 is initialized from the Stage 1 checkpoint rather than from the original pretrained model.

Both recovery stages train all parameters in bf16 with DeepSpeed ZeRO-2 and gradient checkpointing, using a global batch size of 8, AdamW ($\beta = (0.9, 0.999)$, $\varepsilon = 10^{-10}$), cosine scheduling with 200 warmup steps, gradient clipping at 0.05, and learning rate 5×10^{-6}.

Recovery Training Data. For OmniImage-Generation, recovery training uses a mixture of OmniEdit (Wei et al., 2025) and OrionEdit (Jiang et al., 2026). Both recovery stages use the same data mixture. For Zero Shot TTS, recovery training uses the Emilia-Dataset (He et al., 2024). For OmniVideo-Generation, recovery training uses OpenVE-3M (He et al., 2025).

Sampling. All OmniContext images are generated using the same settings across methods: 40 denoising steps, 1024×1024 reference and output resolution, classifier-free guidance with $\gamma = 4$ and a blank negative prompt, and the same fixed seed (42) for all 400 examples. GEdit-Bench follows its released evaluation protocol (50 steps, $\gamma = 4$, and per-example seeds), so its absolute values should not be compared directly with OmniContext scores.

Table 6: **Recovery-stage configurations.**

	Stage 1 Teacher-forced	**Stage 2** On-policy
Weight decay	3×10^{-2}	1×10^{-2}
Training steps	30k	500
Initialization	Pretrained model	Stage 1 checkpoint
Student rollout	—	40 steps, no CFG
Query states	Data-derived interpolants	$N_q = 4$, full trajectory

Scoring. We use the released evaluation code and prompt format of each benchmark without modification. For OmniContext, we follow its VIEScore-style evaluation with GPT-4.1 as the judge.

F What Does Not Work

Before settling on pointwise on-policy regression, we also evaluated guidance-aware Stage 2 variants. We report these experiments because the deployed sampler uses classifier-free guidance, making it natural to ask whether Stage 2 should use the same guided trajectory and branch structure.

Guided rollouts and branch-aware losses. At inference, prediction uses classifier-free guidance

$$v = v^- + \gamma(v^+ - v^-), \qquad \gamma = 4. \tag{26}$$

Our default recovery procedure differs: student rollout, teacher evaluation, and regression all use the conditional branch without CFG. CFG is introduced only during final inference.

We additionally evaluate two branch-aware objectives under a guided student rollout. Following Li et al. (2026), independent branch matching (IBM) minimizes

$$\|v_\theta^+ - v_{\text{full}}^+\|^2 + \|v_\theta^- - v_{\text{full}}^-\|^2, \tag{27}$$

while positive–direction matching (PDM) matches the positive branch together with the guidance direction $d = v^+ - v^-$, using $\lambda = 1$.

Table 7: **Comparison of OPD supervision variants.** We compare unguided OPD with two guidance-based variants on OmniContext. Δ denotes the change in Overall score relative to default OPD.

Variant	Rollout	Supervision	PF ↑	SC ↑	Overall ↑	Δ
Default OPD	unguided	v^+ only	8.418	8.167	**8.185**	—
IBM	guided, γ=4	v^+ and v^-	8.283	7.987	8.025	−0.160
PDM	guided, γ=4	v^+ and d	8.223	8.040	8.021	−0.164

Here Δ is measured relative to the default OPD configuration. Neither tested guidance-aware configuration improves over the default unguided Stage 2 procedure.

These experiments do not isolate the effect of guided rollout from that of branch-aware supervision, because a guided-rollout, positive-only control was not evaluated. The result should therefore be interpreted as a comparison of the complete configurations above rather than as evidence against either component individually.

One possible explanation, consistent with the analysis of Li et al. (2026), is that our setting does not exhibit the same privileged-negative asymmetry. Teacher and student receive the same reference inputs in the negative branch and differ primarily in attention structure. Under such a setting, the

cross-branch compensation targeted by branch-aware objectives may be less pronounced. We leave a controlled decomposition of trajectory guidance and branch supervision to future investigation.

G Additional Results

G.1 Contextualizing the Full-Attention Base Model

Table 8 provides additional context for the full-attention base model by evaluating several recent instruction-based image-editing systems on GEdit-Bench under a common inference and scoring protocol. These systems differ in model scale, training data, architecture, and preferred inference configuration; the table is therefore not intended as a controlled comparison or leaderboard.

All rows are generated and scored by us using the same evaluation pipeline, including the same judge version, prompt format, sampler configuration, resolution, and absence of per-model tuning. The resulting absolute scores may therefore differ from values reported in the corresponding papers, including for our own base model.

Table 8: Recent instruction-based image-editing systems evaluated by us on GEdit-Bench under a common inference and scoring protocol. The table provides additional context for our full-attention base model rather than a controlled leaderboard. *FireRed-Image-Edit is evaluated at version 1.1, while the cited report describes version 1.0.

Model	SC ↑	PQ ↑	Overall ↑
FireRed-Image-Edit-1.1* (Super Intelligence Team et al., 2026)	8.252	7.503	7.762
Qwen-Image-Edit-2511 (our base) (Wu et al., 2025a)	8.068	7.345	7.580
LongCat-Image-Edit (Meituan LongCat Team et al., 2025)	8.041	7.358	7.554
Boogu-Image-0.1-Edit (Chen et al., 2026)	7.784	7.021	7.215
JoyAI-Image-Edit (Song et al., 2026)	7.569	7.271	7.134
FLUX.2 [dev] (Black Forest Labs, 2025)	7.384	7.297	7.073

G.2 Qualitative Examples on OpenVE-Bench

Figure 6 shows representative video-editing results generated by AnchorCache on OpenVE-Bench across different editing scenarios.

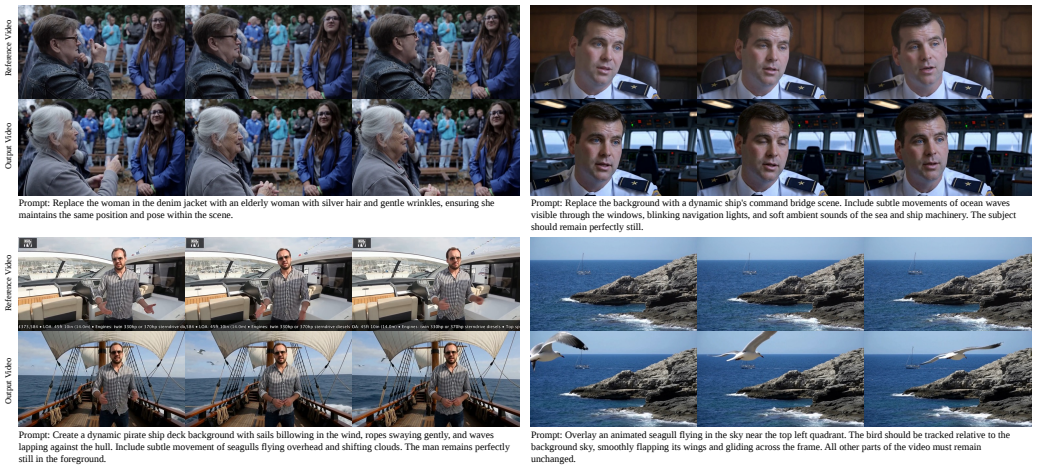

Figure 6: **Qualitative examples on OpenVE-Bench.** Representative video-editing results generated by AnchorCache across different editing scenarios.

G.3 A BATCH OF GEDIT-BENCH EXAMPLES

Figures 8 and 7 show ten contiguous GEdit-Bench examples using the same four-column layout: input, full attention, isolated cache, and AnchorCache. Unlike the selected qualitative examples in the main paper, these figures show a contiguous batch and therefore also include cases for which the three model outputs are visually similar.

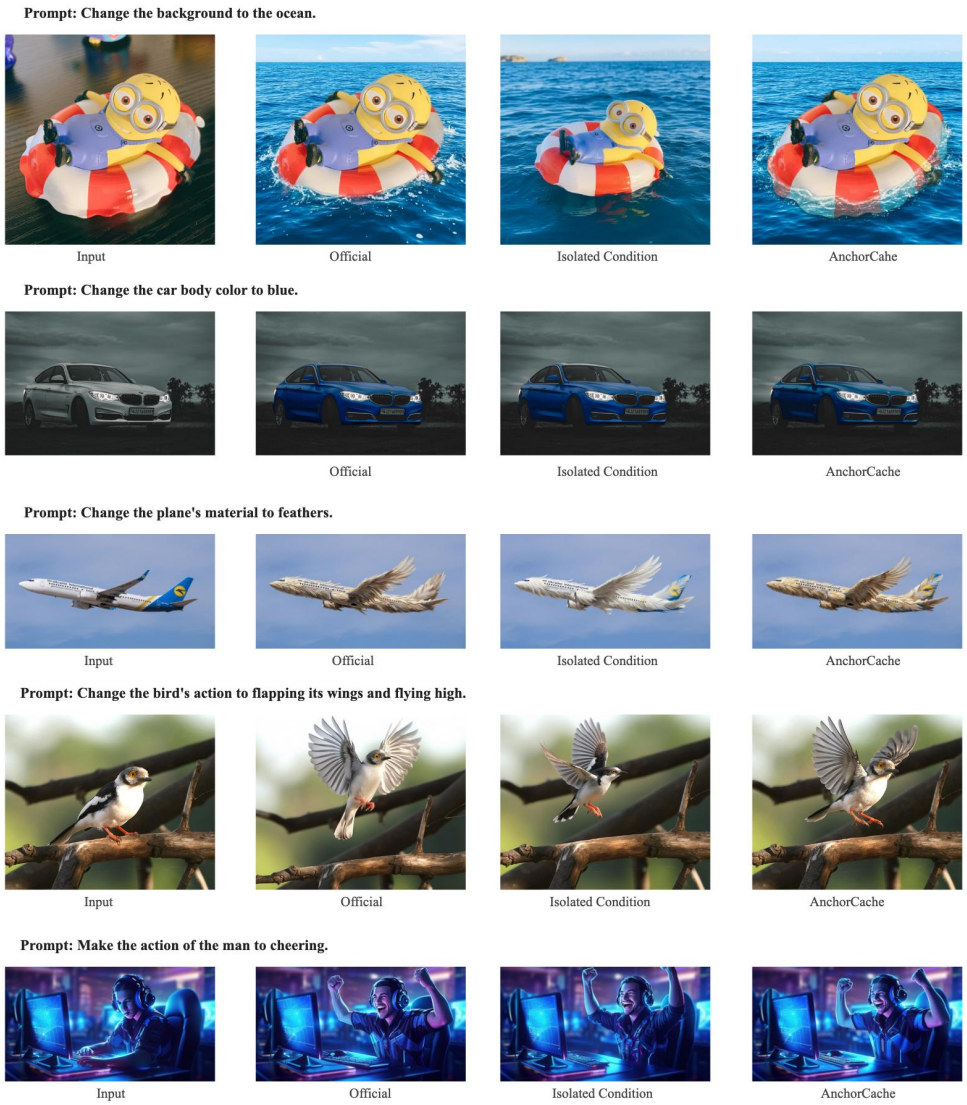

Figure 7: **Qualitative examples on GEdit-Bench, part 1 of 2.** The first five examples from the same contiguous batch.

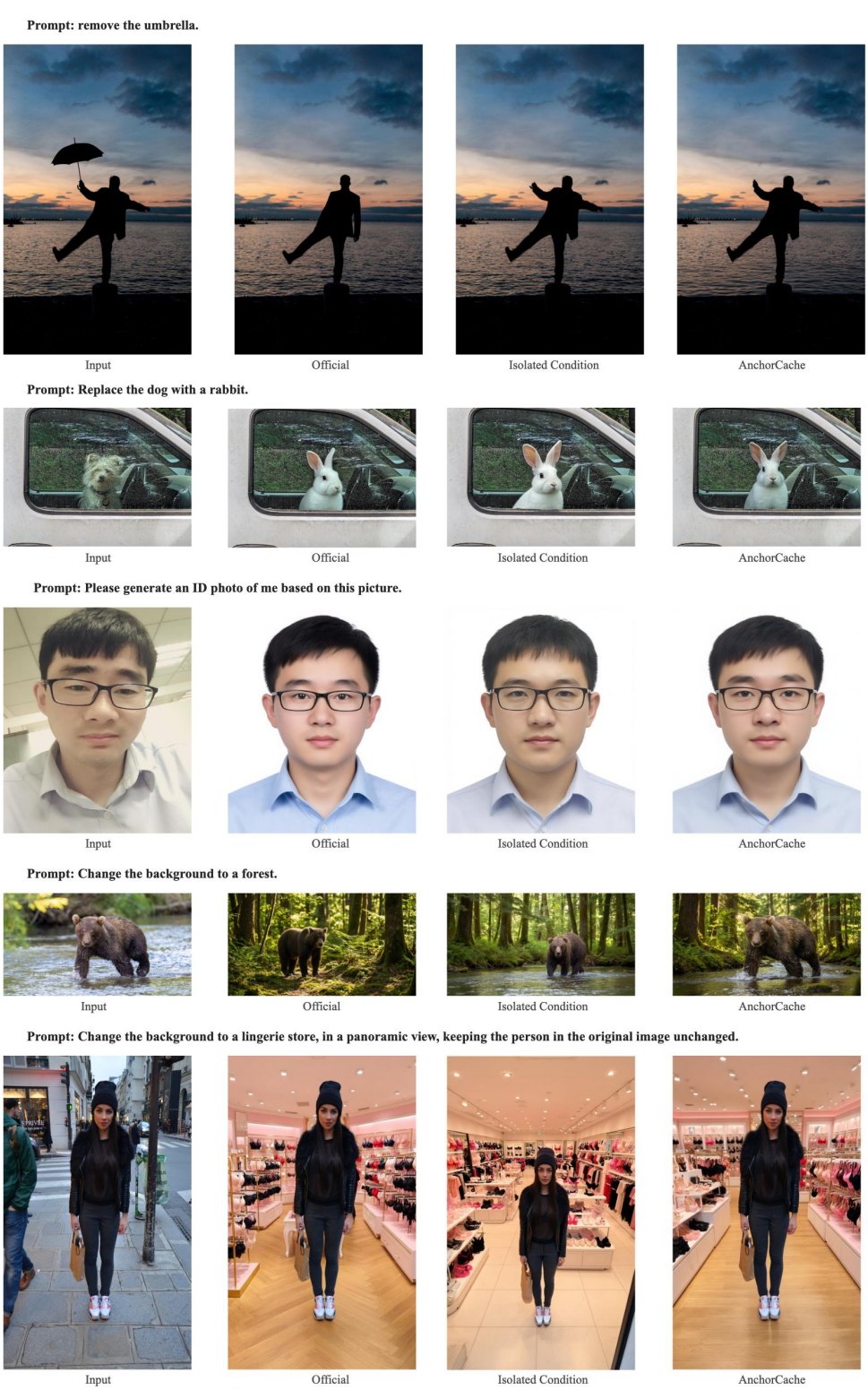

Figure 8: **Qualitative examples on GEdit-Bench, part 2 of 2.** The remaining five examples from the contiguous batch. Columns show the input, full-attention output, isolated-cache output, and AnchorCache output. The editing instruction is shown above each row.