REMIX: EFFICIENT DIFFUSION SERVING WITH ADAPTIVE REUSE AND MODEL MIXTURE

Anonymous authors
Paper under double-blind review

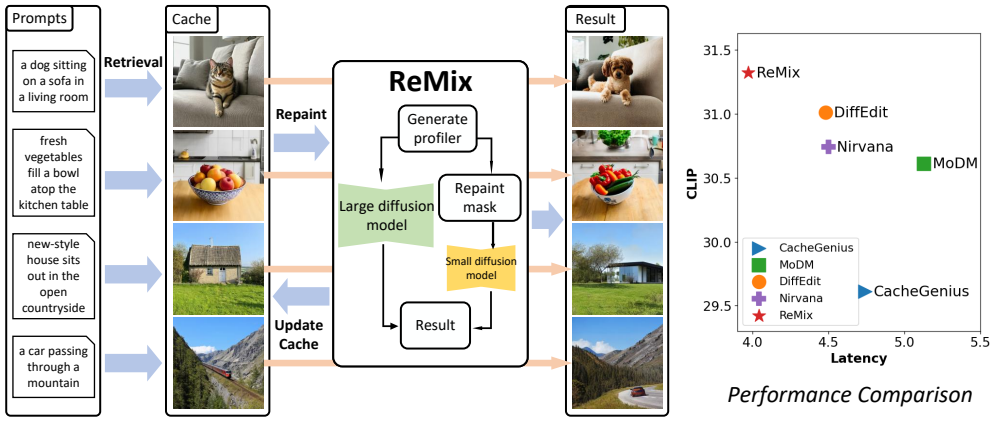


Figure 1: Overview of REMIX, a method for adaptive mixture-of-models reuse in high-quality diffusion serving. By caching large-model outputs and adaptively repainting them with a smaller model, REMIX achieves high-quality generation at substantially lower latency.

ABSTRACT

Serving text-to-image diffusion models online is expensive, because each request runs tens of sequential denoising steps through a large network. Mixture-of-models serving reduces this cost by caching images that a large model has already generated and repainting them with a small model for similar prompts. Existing methods select the repainting strength by fixed hand-crafted rules and repaint the entire image, which miscalibrates the strength and overwrites content that was already correct. We propose REMIX, an efficient diffusion serving framework that reuses cached images and generates with a mixture of models. REMIX profiles the denoising trajectory of each cached generation to select a per-request repainting start step and repaints under a graduated change map that expands from the high-change regions to the full image, leaving remaining pixels at the fidelity of the cached image. On COCO and DiffusionDB, REMIX achieves the best CLIP, FID, IS, and PickScore among all serving methods with the lowest mean latency. Code is available at `https://anonymous.4open.science/r/ReMix-D2AF`.

1 INTRODUCTION

Diffusion models (Ho et al., 2020; Rombach et al., 2022; Saharia et al., 2022) generate high-quality images and are widely used in graphic design, content creation, and synthetic data generation. Commercial systems such as Adobe Firefly (Adobe, 2023) now expose them to end users at scale. In production, these models must handle large numbers of diverse, concurrent generation requests. Each generation runs tens of sequential denoising steps (Song et al., 2021) through a large neural network, typically spending several seconds of GPU computation per image. Reducing the per-request inference cost while preserving generation quality is thus a pressing challenge for deploying diffusion models as online services.

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.

Existing work improves diffusion serving efficiency by caching data during generation, along two orthogonal axes. In-model caching stores the latent produced at each denoising step and reuses it across steps, since consecutive latents change only slightly, saving substantial computation (Wimbauer et al., 2024; Ma et al., 2024b;a). Mixture-of-models reuse instead caches the images generated by a high-quality large model and repaints them with a small model for similar new requests (Agarwal et al., 2024; Xia et al., 2026). The two address distinct sources of overhead. In-model caching reduces the cost of each denoising step, while mixture-of-model reuse lowers generation latency by cutting the number of steps each request needs to execute. Furthermore, by leveraging images instead of intermediate computations, the mixture-of-model reuse technique generalizes to any pair of models without changes to the model architecture.

Despite these advances, mixture-of-models reuse methods face two fundamental limitations. First, existing methods select the number of noise steps for repainting through heuristic rules (Xia et al., 2026; Agarwal et al., 2024; Cui et al., 2025). For instance, they partition the similarity between the cached and incoming prompts into discrete bins and map each bin to a predetermined step count. This heuristic mapping ignores the intrinsic structure of the diffusion process, where early steps determine global layout, middle steps refine semantics, and late steps add fine details. It also cannot adapt to varying model architectures, generation tasks, or prompt distributions without manual retuning. Second, all existing methods perform full-image repainting. The small model redraws every pixel of the image uniformly, regardless of how much each region actually needs to change. This wastes computation on regions that require no modification, but more importantly, it destroys the high-quality content already produced by the large model in those regions. The small model, rather than inheriting and refining the large model's output only where needed, must reconstruct the entire image from scratch given a noisy starting point.

We propose REMIX, as Figure 1 shows, a method for efficient diffusion serving. Like other mixture-of-models methods, REMIX caches images generated by a large model and repaints them with a small model, saving inference time. To address the limitations above, REMIX introduces two techniques, illustrated in Figure 2. Profiling-driven adaptive step selection analyzes the denoising trajectory of each large model generation through two per-step signals, the noise-prediction delta and the CLIP alignment score, which respectively indicate when the image structure stabilizes and when semantic quality saturates. These signals locate two boundary steps that delimit the semantic planning stage, over which REMIX fits a piecewise CLIP-to-step mapping. At serving time, REMIX queries this mapping with the semantic gap to recover a per-sample repainting start step. Graduated Repainting computes a change map from the noise-prediction divergence between the original and new prompts, then applies a threshold that decays linearly to zero so that the binary mask expands from core-change regions to the full image. Outside the mask, a shared-noise anchor state preserves unmodified regions at pixel fidelity, avoiding the drift of DDIM inversion.

Our contributions are as follows:

- We introduce profiling-driven adaptive step selection, a technique that profiles the denoising trajectory of each large model generation to derive a per-sample repainting start step, replacing the hand-crafted heuristic step mappings of prior work.
- We propose Graduated Repainting, a soft-mask repainting strategy that computes a change map from the noise-prediction divergence and applies a progressively decaying threshold to protect unmodified image regions while enabling precise targeted editing.
- We evaluate REMIX on COCO and DiffusionDB. It reaches the best CLIP, FID, IS, and PickScore among all serving methods while cutting the mean serving latency by 52.3% and 26.3% against full large-model generation. Its CLIP loss is up to 59% smaller than that of the runner-up baseline, and its PickScore loss up to 19% smaller.

2 RELATED WORK

2.1 MIXTURE-OF-MODELS REUSE FOR DIFFUSION SERVING

Generative models are widely deployed as services (Adobe, 2023; OpenAI, 2023), and work on efficient diffusion serving has shifted from accelerating a single model to coordinating heterogeneous models. Early methods assign the critical denoising steps of a generation to the large model and

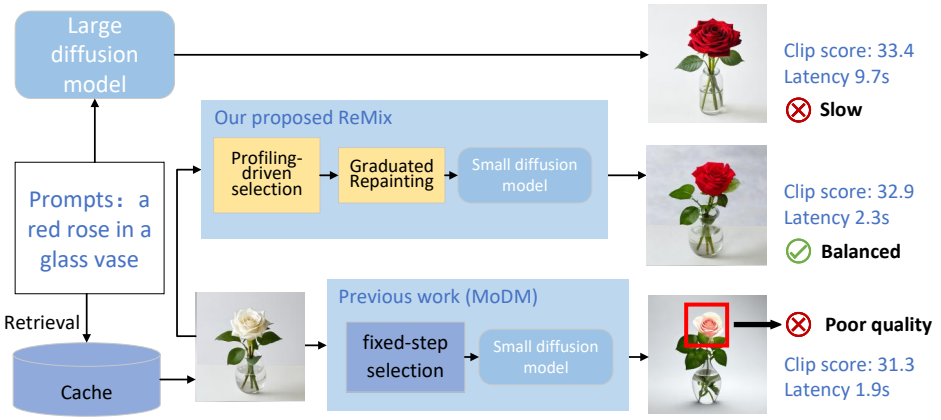


Figure 2: Compared with MoDM, which adopts fixed-step selection to feed intermediate latent representations into a small diffusion model, our REMIX employs profiling-driven step selection and constructs a repaint mask to guide the lightweight model. Direct generation by the large model yields a better image but takes 9.7s. REMIX keeps the vase intact while changing the flower color as the new prompt requires, at a semantic alignment close to the large model and a latency of 2.3s. MoDM does not fully repaint the image according to the new semantics.

the remaining steps to a lightweight model (Pan et al., 2025; Yan et al., 2024; Xie et al., 2025). A parallel line of work exploits structure across prompts and divides into request routing and reuse. Routing methods build a cascade of models and train a classifier that labels each request by prompt complexity. Hard requests are sent to the large model and easy requests to the small model (Ahmad et al., 2025; Xin et al., 2025; Li et al., 2025; Lu et al., 2024). Reuse-based designs are motivated by analyses of DiffusionDB and of Adobe's production traffic, which report that online prompts contain many semantic near-duplicates (Agarwal et al., 2024; Wang et al., 2023b). Early work reuses the latents of frequent prompts (Agarwal et al., 2024). When a similar prompt arrives, the system selects the latent at a corresponding step by similarity and repaints the image from that step. Later work shows that a small model can perform this repaint while saving resources and preserving generation quality (Xia et al., 2026; Cui et al., 2025). These methods set the repainting strength by hand-crafted rules, so the repainted images do not fully exploit the cached content. REMIX improves this pipeline through profiling-driven step selection and Graduated Repainting.

2.2 IMAGE INPAINTING AND EDITING

Diffusion-based inpainting and editing modify a localized region of an image, and existing methods fall into mask-guided and text-guided families (Huang et al., 2025). Mask-guided methods require the user to specify the edit region, which is converted into a binary mask that confines generation to the target area (Avrahami et al., 2022; Lugmayr et al., 2022). Follow-up work targets the interference between the edited region and its background, balancing background preservation against prompt adherence through noise optimization and progressive latent guidance (Lugmayr et al., 2022; Wang et al., 2023a; Manukyan et al., 2025; Gong et al., 2026). Text-guided methods infer the edit region implicitly from the prompt and the source image. Cross-attention control preserves the original layout during editing (Hertz et al., 2023), instruction-tuned models follow natural-language edit commands (Brooks et al., 2023), and DiffEdit contrasts noise predictions conditioned on the source and target prompts to derive an edit mask (Couairon et al., 2023). These methods are designed for interactive single-image editing. Text-guided approaches depend on image inversion or cross-attention computation. Recent one-step methods remove this cost but require additional training, which restricts their deployment in a serving system (Nguyen et al., 2025). The inpainting methods described above are also tuned for local modifications, whereas in cross-prompt reuse the source image and the incoming prompt can differ in content and layout. Graduated Repainting instead exploits the cached generation information available in mixture-of-models reuse. It adopts DiffEdit's noise-prediction divergence with two adaptations for serving. The threshold decays across denoising

steps rather than staying at a fixed binarization value, so the mask expands from the core-change regions to the full image. Unmodified regions are anchored to a shared noise tensor, which avoids the cost and drift of DDIM inversion.

2.3 Efficient Diffusion Inference

Beyond mixture-of-models reuse, efficient diffusion inference advances along two directions. One line reduces the number of denoising steps, and the other reduces the cost of a single generation. Early step-reduction methods shorten the sampling trajectory through continuous-time formulations and fast solvers (Kong & Ping, 2021) and through early termination of the diffusion process (Lyu et al., 2022). A fixed reduced schedule allocates the same number of steps to every prompt, so later work targets adaptive execution. These methods predict the number of denoising steps from prompt complexity and vary the network's computational effort across diffusion timesteps (Zhang et al., 2025; Moon et al., 2023). Reducing the cost of a single generation relies on caching data produced during denoising. Early schemes reuse attention maps or residual blocks (Hunter et al., 2025; Wimbauer et al., 2024), and subsequent work reuses UNet features across adjacent timesteps (Ma et al., 2024b). Recent methods refine the reuse granularity and cache policy through token-wise, adaptive, trajectory-aware, and speculative caching (Zou et al., 2024; Bu et al., 2026; Chu et al., 2025; Liu et al., 2025). These methods improve the efficiency of each individual generation, but they are harder to deploy and often require changes to the model architecture. They also exploit only the optimization space within a single model, and so overlook the benefit of mixture-of-models serving, which lowers GPU memory demand by serving cache hits with a small model.

3 Method

ReMix accelerates text-to-image diffusion serving by reusing cached images across similar requests. When a new request arrives, ReMix determines whether it hits the cache by the similarity between the prompt and the cached images. On a miss, a large diffusion model generates the image and profiles its denoising trajectory. On a hit, ReMix derives the repainting step and strength from the profile metadata, and builds a mask from the semantic difference between the original and current prompts to guide the repainting process.

3.1 Profiling-driven Adaptive Step Selection

Upon receiving a request, ReMix retrieves a similar cached image and repaints it by adding noise at a preset strength, then denoising it with the diffusion model conditioned on the new prompt. This procedure skips part of the denoising steps and costs less than generating from scratch. The repainting strength depends on how many noise steps are added. Prior methods decide this number through hand-crafted branching rules, such as mapping a similarity bin to a fixed step count. These rules rely on manual tuning and cannot reliably control the strength.

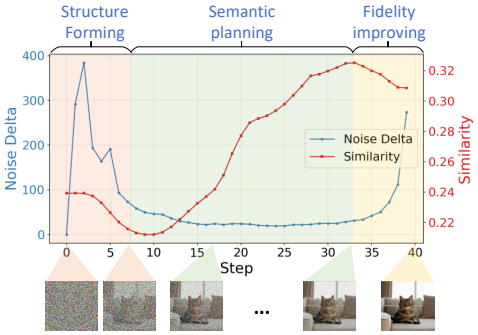


Figure 3: When generating images with the Stable Diffusion 3.5 model, the Similarity and Noise-delta curves at each step divide the denoising process into three stages: structure forming, semantic planning, and fidelity improving.

ReMix instead adopts profiling-based step selection. When a request misses the cache, the large diffusion model generates the image while saving its intermediate states, from which we profile two per-step signals. The first is the noise-prediction delta

$$\Delta_i = \|\epsilon_\theta(x_i, i, c) - \epsilon_\theta(x_{i-1}, i-1, c)\|_2, \quad (1)$$

which measures the ℓ_2 change in predicted noise between consecutive steps. The second is the cosine similarity between the partially denoised latent and the prompt, obtained by periodically decoding the saved latents.

The motivation for profiling these two signals stems from an observation about the denoising process. As shown in Figure 3, a denoising trajectory can be divided into three stages by the noise delta and the : structure forming, semantic planning, and fidelity improving. In the structure forming stage, the noise delta changes rapidly as the model establishes the image layout. In the semantic planning stage, the similarity rises steadily as the semantic content sharpens. In the fidelity improving stage, the similarity saturates while the noise delta rises again, as the final steps refine full-image quality. This observation implies that, when a cache hit triggers repainting, the start step should lie in the semantic planning stage, so as to reuse the layout built by the large model while steering the semantic content toward the new prompt.

REMIX analyzes the trajectory to locate two boundary steps via change-point detection. The minimum stable step $s_{\min}$ is the change point of the noise-delta curve, where Δ_i transitions from a rapidly decreasing regime to a stable low-amplitude regime. The maximum effective step $s_{\max}$ is the change point of the CLIP curve, where the similarity transitions from a rising regime to a saturating regime. Each change point is the breakpoint of a two-segment linear fit that minimizes the total squared residual,

$$s_{\min} = \arg\min_{s} \ \mathrm{SSE}(\Delta_{1:s}) + \mathrm{SSE}(\Delta_{s+1:T}), \tag{2}$$

$$s_{\max} = \arg\min_{s \geq s_{\min}} \ \mathrm{SSE}(\mathrm{CLIP}_{s_{\min}:s}) + \mathrm{SSE}(\mathrm{CLIP}_{s+1:T}), \tag{3}$$

where $\mathrm{SSE}(\cdot)$ is the sum of squared residuals of the best linear fit to the sequence, and $\Delta_{a:b}$ and $\mathrm{CLIP}_{a:b}$ denote the corresponding subsequences. Across all profiled trajectories we observe that the similarity is non-decreasing over $[s_{\min}, s_{\max}]$ in the vast majority of cases, so the mapping can be inverted. To guarantee invertibility regardless of local fluctuations, REMIX fits the sampled similarities with a monotone piecewise-linear function

$$g(s) = a_k\, s + b_k, \quad s \in [s_k, s_{k+1}], \quad k = 0, 1, \ldots, M-1, \quad a_k \geq 0, \tag{4}$$

where $s_0 = s_{\min} < s_1 < \cdots < s_M = s_{\max}$ are the breakpoints. The non-negative slope constraints make g monotone by construction, so its inverse is always well-defined. On a cache hit, REMIX reads the stored score of the cached image under its source prompt, $\mathrm{CLIP}_{\text{orig}}$, and computes its score under the new prompt, $\mathrm{CLIP}_{\text{new}}$. The gap $\mathrm{CLIP}_{\text{orig}} - \mathrm{CLIP}_{\text{new}}$ measures how much semantic content must change. REMIX steps back along the trajectory by this gap, scaled by κ so that the cache-hit threshold τ_{clip} maps to the earliest step $s_{\min}$, and recovers the start step as

$$s_{\text{start}} = g^{-1}\Big(\mathrm{CLIP}_{\text{orig}} - \big(\mathrm{CLIP}_{\text{orig}} - \mathrm{CLIP}_{\text{new}}\big) \cdot \kappa\Big), \quad \kappa = \frac{\mathrm{CLIP}_{\text{orig}} - g(s_{\min})}{\mathrm{CLIP}_{\text{orig}} - \tau_{\text{clip}}}. \tag{5}$$

A larger gap yields an earlier start step and thus a stronger repainting.

Each cache entry stores the generated image, its source prompt, and a compact profile. The profile records the minimum stable step $s_{\min}$, the maximum effective step $s_{\max}$, the score $\mathrm{CLIP}_{\text{orig}}$ of the cached image under the source prompt, and a piecewise-linear map g over $[s_{\min}, s_{\max}]$ that describes how the similarity evolves along the trajectory. Because the image dominates the storage footprint, this metadata adds a negligible overhead of only a few hundred bytes per entry.

On a cache hit, the scheduler reads the profile and sets the repainting start step to s_{start}. The repainting strength is then

$$\text{strength} = (T - s_{\text{start}}) \,/\, T, \tag{6}$$

where T is the total number of denoising steps. A larger s_{start} yields fewer repainting steps, as prompts closer in semantics to the cached prompt require less modification. Unlike heuristic step-count tables, this approach is data-driven, adapts to each image's denoising dynamics, and requires no model-specific calibration.

3.2 GRADUATED REPAINTING

We further find that full-image repainting with the small diffusion model distorts the background, which damages the parts of the high-quality cached image that do not need modification. REMIX therefore proposes the mask-guided Graduated Repainting technique, which localizes the change and edits only there. We encode the cached image into a clean latent z_0, draw a shared noise $\boldsymbol{\eta}$, and

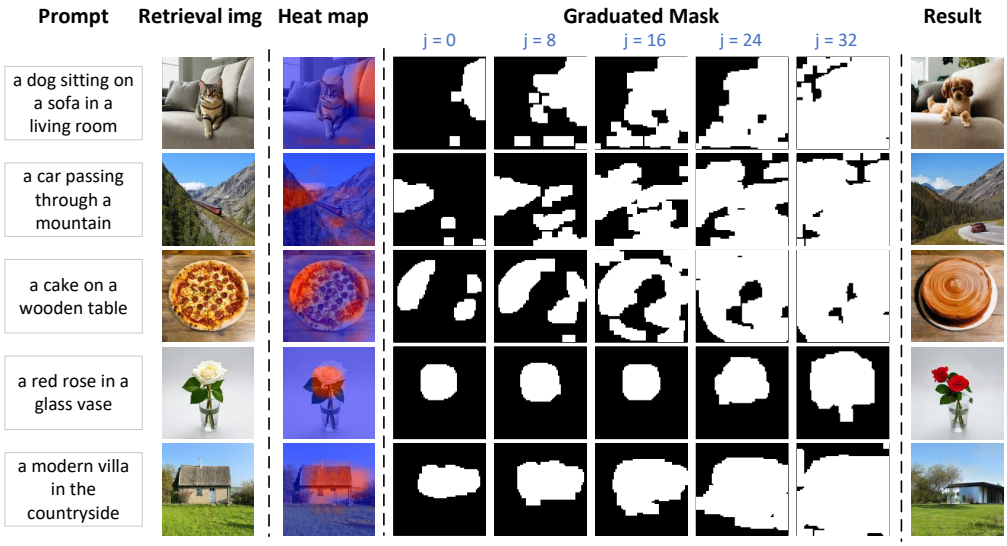


Figure 4: Visualization of Graduated Repainting on one example. The heatmap shows the change map M_{diff}, the element-wise noise-prediction divergence between the original and new prompts, with brighter regions indicating stronger semantic change. The binary masks B_j at successive repainting steps show how the graduated threshold ρ_j expands the mask from core-change regions to the full image as denoising proceeds.

noise z_0 to the start timestep $t_0 = T - s_{\text{start}}$ using $\boldsymbol{\eta}$. Two UNet passes conditioned on the original and new prompts yield noise predictions whose element-wise ℓ_2 difference forms the change map

$$M_{\text{diff}}(h, w) = \|\boldsymbol{\epsilon}_{\text{new}}(h, w) - \boldsymbol{\epsilon}_{\text{orig}}(h, w)\|_2, \tag{7}$$

normalized to $[0, 1]$ and smoothed by max pooling. Large values mark regions whose content must change.

Binarizing the change map at a fixed threshold under- or over-edits and creates boundary artifacts. We instead use a graduated threshold that decays linearly from an initial value to zero over the $K = s_{max} - s_{\text{start}}$ repainting steps, which we index by $j = 0, \ldots, K-1$ with timesteps $t_j = K - j$,

$$\rho_j = \rho_{\text{init}} \cdot \left(1 - \frac{j}{K-1}\right), \tag{8}$$

where $\rho_{\text{init}} = \mu(M_{\text{diff}}) + \sigma(M_{\text{diff}})$ adapts to the change-map statistics. At step j, the binary mask is

$$B_j = \text{MaxPool}_{3\times 3}\big(\mathbb{1}[M_{\text{diff}} > \rho_j]\big), \tag{9}$$

so the mask expands monotonically from core-change regions to the full image as denoising proceeds, as shown in Figure 4.

To preserve pixel fidelity outside the mask, we maintain an anchor state, which is the clean latent noised to any timestep with the same shared noise,

$$z_t^{\text{anchor}} = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\eta}. \tag{10}$$

At step j we denoise inside the mask with classifier-free guidance and blend the result with the anchor outside it,

$$z_{t_{j+1}} = B_j \odot z_{t_{j+1}}^{\text{denoised}} + (1 - B_j) \odot z_{t_{j+1}}^{\text{anchor}}. \tag{11}$$

Because the anchor shares the noise of the initial noising, masked-out regions reproduce the cached image exactly. As ρ_j decays to zero, the mask covers the entire image and the final output fully aligns with the new prompt.

This graduated masking avoids the boundary seams of a static binary mask. A fixed threshold keeps a hard boundary between the denoised and anchored regions, which produces visible seams

at the mask edge. By contrast, the decaying threshold lets the boundary sweep outward across the repainting steps, so each region transitions gradually from the anchored state to the denoised state. Because the anchored and denoised regions share the same noise level at every step, their boundary differs only in content, and the 3×3 max pooling further softens it. At the final step the mask covers the whole image and the anchor is fully released, which leaves no seam in the output.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate on two standard text-to-image generation benchmarks: the COCO 2017 validation set and DiffusionDB (Wang et al., 2023b). For each dataset, we follow the protocol of MoDM (Xia et al., 2026). COCO provides diverse natural-image prompts with ground-truth reference images for FID computation; DiffusionDB contains user-generated prompts spanning a wide range of styles and concepts, which tests generalization to open-ended text inputs.

Models. We use Stable Diffusion 3.5 Medium (AI, 2024) as the large model and Stable Diffusion v1.5 (AI, 2022) as the lightweight repainting model. All models operate at 512×512 resolution with $T = 50$ DDIM steps and classifier-free guidance scale 7.5. The cache is pre-populated with images generated by SD3.5 from the training prompts of each dataset. A request hits the cache when the CLIP score of the cached image under the incoming prompt exceeds the threshold $\tau_{\text{clip}} = 0.26$.

Baselines. We compare six systems: (i) **SD3.5** (AI, 2024), large-model generation for every request, which serves as the quality reference; (ii) **CacheGenius** (Cui et al., 2025), an in-model caching system that reuses intermediate computation across similar requests; (iii) **DiffEdit** (Couairon et al., 2023), mask-based editing with a static binary mask from the noise-prediction difference, which isolates masking without the graduated threshold; (iv) **MoDM** (Xia et al., 2026), the state-of-the-art mixture-of-models method, which maps prompt similarity to a repainting step count through a hand-crafted table and repaints the full image with SD1.5, and which shares our reuse setting but enables neither component; (v) **Nirvana** (Agarwal et al., 2024), reuse driven by image-similarity matching between requests and cached entries; and (vi) REMIX, our method, which profiles the cached image's denoising trajectory to select the repainting start step and repaints under a graduated semantic mask. Single-component variants appear in the ablation study (Section 4.4).

Metrics. We report five image quality metrics. We measure text and image alignment with CLIP Score (Hessel et al., 2021) and PickScore (Kirstain et al., 2023), distributional similarity to real images with the Fréchet Inception Distance (FID) (Heusel et al., 2017), diversity and visual quality with Inception Score (IS) (Szegedy et al., 2016), and perceptual similarity with LPIPS (Zhang et al., 2018). For serving performance, we report the mean end-to-end latency.

Implementation. All experiments are conducted on two NVIDIA A100-40GB GPUs. Profiling introduces a 4.8% time overhead to large-model generation on the critical path by recording intermediate latents. The remaining analysis incurs overhead equivalent to 109.4% of one generation and runs in an offline thread without blocking responses. This analysis cost is amortized over the serving lifetime and excluded from latency measurements, as detailed in Appendix B.3. All cache-based baselines share the same cache hit criterion, which keeps the cache hit rate constant across methods and isolates the effect of the repainting method. The cache size is set to 10% of the total dataset size, and all baselines use a first-in first-out (FIFO) replacement policy. Cache retrieval latency is included in end-to-end measurements and remains negligible, at less than 50 ms per query.

4.2 IMAGE QUALITY AND SERVING LATENCY

Table 1 reports the image quality metrics and mean serving latency for all methods on both datasets. SD3.5, as the large-model reference, is separated from the serving methods by a horizontal rule.

Text-image Alignment. We evaluate the text-image alignment performance of our method using the CLIP score and PickScore. REMIX attains the best text-image alignment among all serving

Dataset	Method	CLIP ↑	PickScore ↑	FID ↓	IS ↑	LPIPS ↓	Mean Lat. (s) ↓
COCO	SD3.5	31.80	22.43	110.13	13.42	0.31	8.78
	CacheGenius	29.61	21.12	<u>117.01</u>	<u>14.85</u>	0.746	4.74
	MoDM	30.61	<u>21.54</u>	117.84	11.97	0.747	**4.25**
	DiffEdit	<u>31.01</u>	21.34	113.23	11.85	<u>0.737</u>	4.48
	Nirvana	30.74	21.35	128.95	11.84	0.770	4.53
	REMIX	**31.36**	**21.67**	**111.41**	**18.29**	**0.281**	<u>4.42</u>
DiffusionDB	SD3.5	33.17	21.26	129.20	6.64	0.30	8.79
	CacheGenius	30.69	19.95	<u>131.85</u>	<u>8.27</u>	<u>0.264</u>	6.68
	MoDM	32.19	20.66	141.01	7.62	0.266	6.54
	DiffEdit	<u>32.21</u>	<u>20.69</u>	134.95	7.87	0.719	6.59
	Nirvana	31.34	20.18	136.44	7.69	0.733	**6.19**
	REMIX	**32.78**	**20.80**	**127.13**	**11.13**	**0.279**	<u>6.61</u>

Table 1: Image quality and serving latency comparison on COCO and DiffusionDB. Bold and underlined indicate the best and second-best among serving methods, respectively. ↑ denotes higher is better and ↓ denotes lower is better.

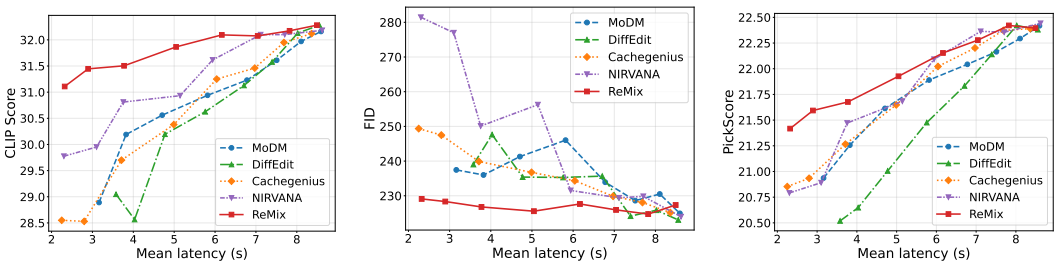
methods on both datasets. On COCO and DiffusionDB it retains 98.6 to 98.8% of the full model's CLIP score and 96.6 to 97.8% of its PickScore. Its CLIP loss is up to 59% smaller than that of the runner-up baseline, and its PickScore loss up to 19% smaller. This result follows from how the profiler sets the repainting start step. REMIX caches the CLIP score at every denoising step, and it uses that trace to choose the start step at which the cached image aligns most fully with the new prompt.

Generated Image Distribution. We evaluate the generated image distribution using FID and IS, and perceptual similarity via LPIPS. REMIX lowers FID to 102.09 on COCO and 116.19 on DiffusionDB, 8.3% and 6.2% below CacheGenius, the strongest baseline on this metric, and raises the Inception Score by 23.2% and 34.5% over the same method. This result follows from Graduated Repainting. The change map comes from the noise-prediction divergence between the cached and the new prompt, and its threshold decays to zero so the mask grows from the core-change regions to the whole image. This growth protects the cached content while prompt semantics are formed in the early steps and blends the edited boundaries into the original image later. As a result REMIX restores LPIPS to 0.281 and 0.279, matching the SD3.5 level, while every other baseline falls to 0.713 or above. DiffEdit, the strongest baseline on text-image alignment, does not lead on distribution, and CacheGenius, which leads on distribution, holds the weakest alignment of any method. REMIX is the only method that leads on both families, and it balances semantic alignment against image quality instead of exchanging one for the other.

Latency. REMIX is the fastest serving method on COCO and the second fastest on DiffusionDB. It cuts the mean latency of full large-model generation by 52.3% and 26.3%, reaching 4.19s and 6.48s, and it is 18.3% faster than MoDM on COCO. This result follows from the start step the profiler selects. A repaint runs $T - s_{\text{start}}$ steps, and the profiler sets s_{start} from the semantic gap, so a prompt close in meaning to its cached entry starts late and finishes in few steps. Graduated Repainting adds no latency of its own, because the pixels outside the mask come from closed-form noising of the cached image rather than from extra network passes.

4.3 PARETO FRONT COMPARISON

To test how sensitive REMIX is to the cache hit threshold, we sweep τ_{clip} from 0.23 to 0.30, which moves the hit rate from 0.90 to 0.05. A lower threshold admits more cache matches, so a larger share of requests enters the repainting path with a cached image that is less similar to the incoming prompt. This is the regime of high hit rates and low latency, and it is where REMIX delivers its largest advantage, as Figure 5 shows. It does so because the change map confines the edit to the regions that the new prompt actually affects, so that even a weakly matched cache entry can be repainted to follow the new prompt. Across the sweep REMIX leads on CLIP, FID, and PickScore,



(a) CLIP score (b) FID (c) Pick score

Figure 5: Pareto-front comparison evaluated by different image quality metrics.

Dataset	Baseline	Step Sel.	Grad. Mask	CLIP ↑	FID ↓	IS ↑	PickScore ↑	Lat. (s) ↓
COCO	(a)	×	×	30.61	117.84	11.97	21.54	**4.25**
	(b)	×	✓	<u>31.16</u>	113.70	12.15	21.54	<u>4.28</u>
	(c)	✓	×	31.03	**110.65**	<u>12.48</u>	<u>21.66</u>	4.38
	(d)	✓	✓	**31.36**	<u>111.41</u>	**18.29**	**21.67**	4.42
DiffusionDB	(a)	×	×	32.19	141.01	7.62	20.66	**6.54**
	(b)	×	✓	32.50	<u>128.71</u>	<u>9.81</u>	20.61	<u>6.56</u>
	(c)	✓	×	<u>32.55</u>	129.53	6.64	<u>20.79</u>	6.58
	(d)	✓	✓	**32.65**	**127.13**	**11.13**	**20.80**	6.61

Table 2: Ablation on the two components of REMIX on COCO and DiffusionDB. Bold and underlined indicate the best and second-best values per column within each dataset.

and its advantage narrows only as the hit rate approaches 0.05 and reuse effectively stops. The method therefore holds a stable advantage as the cache hit rate varies.

4.4 ABLATION STUDIES

We isolate the contribution of each component of REMIX with a 2×2 study over its two techniques, profiling-driven step selection and Graduated Repainting, on both COCO and DiffusionDB. Table 2 reports four variants: (a) MoDM, which uses heuristic step selection and full-image repainting; (b) Graduated Repainting under MoDM's step selection; (c) profiling-driven step selection with full-image repainting; and (d) the full REMIX. The two components act on different properties of the output. Profiling-driven step selection lowers FID in all four comparisons, by at least 7.2 points on COCO and 12.1 on DiffusionDB. Graduated Repainting governs diversity instead, raising the Inception Score to 16.42 on COCO and 10.34 on DiffusionDB against 12.48 and 6.64 for the profiler alone. This gain depends on the start step the profiler provides, since under the heuristic step the mask alone lowers the DiffusionDB Inception Score from 7.62 to 6.57 by suppressing regions that still carry prompt-relevant content. The two components are also not additive on DiffusionDB, where their combined FID gain of 24.36 points falls short of the 32.83 their separate gains predict. The profiler bounds how far the repainting trajectory travels before it satisfies the new prompt, and the mask bounds how many pixels the residual drift can reach.

5 CONCLUSION

We address two challenges in mixture-of-models diffusion serving: heuristic step selection that ignores per-sample denoising dynamics, and uniform full-image repainting that degrades high-quality cached content. Our solution introduces REMIX, which combines profiling-driven adaptive step selection, a technique that derives per-sample repainting start steps from each generation's denoising trajectory, and Graduated Repainting, a technique that uses a change map with a progressively decaying threshold to restrict repainting to semantically altered regions while preserving unmodified content at pixel fidelity. Experiments on COCO and DiffusionDB show that REMIX achieves the best CLIP, FID, IS, and PickScore among all serving methods and surpasses the large model's FID and IS on both datasets.

ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. No animal experimentation and no research on human subjects were involved. We use the COCO and DiffusionDB prompt sets and their publicly released images, which are distributed under terms that permit research use, and we comply with those terms. The cached generations are produced by our own models and contain no private or personally identifiable information, and no experiment in this work raises privacy or security concerns. Text-to-image models can reproduce biases present in their training data, and our method leaves the base models and their training data unchanged, so it neither introduces biases of its own nor claims to remove existing ones. To avoid drawing conclusions from a single prompt distribution, we report results on two datasets, and we evaluate every method under the same cache protocol so that no method is favored by the experimental setup.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we have made the following efforts. (1) We will release our code of REMIX. (2) We evaluate on COCO and DiffusionDB, both publicly available, so that all quality and latency numbers can be reproduced without private data. (3) Section 4 reports the experimental setup and the evaluation protocol, including the cache size, the replacement policy, and the shared cache hit criterion that keeps the hit rate constant across methods. (4) Appendix A provides the full method and system details, covering the notation, the data flow, both algorithms, and a storage and complexity analysis. (5) Appendix B.3 reports the profiling overhead and its placement outside the critical path, so that the latency measurements can be reproduced. We believe these measures will enable other researchers to reproduce our work and further advance the field.

AI USE STATEMENT

In this work, we used generative AI tools for language editing and for reorganizing the presentation of the experimental results. We have not used generative AI tools for proposing the research idea, designing the method, implementing the code, or running the experiments, and the remaining tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools for translating manuscript passages written in Chinese into English and for checking terminology consistency across sections. We have reviewed all AI-assisted work. All experimental results were produced by our own implementation, and no reported number was generated or altered by an AI tool. The authors verified every quantitative claim in the text against the recorded experimental logs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Adobe. Adobe unleashes new era of creativity for all with the commercial release of generative ai. PDF press release, sep 2023. URL https://news.adobe.com/assets/downloads/pdfs/2023/09/091323adobecommercialfirefly.pdf.

Shubham Agarwal, Subrata Mitra, Sarthak Chakraborty, Srikrishna Karanam, Koyel Mukherjee, and Shiv Kumar Saini. Approximate caching for efficiently serving text-to-image diffusion models. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pp. 1173–1189, 2024.

Sohaib Ahmad, Qizheng Yang, Haoliang Wang, Ramesh K Sitaraman, and Hui Guan. Diffserve: Efficiently serving text-to-image diffusion models with query-aware model scaling. *Proceedings of Machine Learning and Systems*, 7, 2025.

Stability AI. Stable diffusion v1.5. 2022. https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5.

Stability AI. Stable diffusion 3.5 medium. 2024. https://huggingface.co/stabilityai/stable-diffusion-3.5-medium.

Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18187–18197, 2022. doi: 10.1109/CVPR52688.2022.01767.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18392–18402. IEEE, 2023.

Jiazi Bu, Pengyang Ling, Yujie Zhou, Yibin Wang, Yuhang Zang, Dahua Lin, and Jiaqi Wang. Dicache: Let diffusion model determine its own cache. In *International Conference on Learning Representations*, volume 2026, pp. 73778–73803, 2026.

Huanpeng Chu, Wei Wu, Guanyu Feng, and Yutao Zhang. Omnicache: A trajectory-oriented global perspective on training-free cache reuse for diffusion transformer models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16302–16312. IEEE, 2025.

Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3lge0p5o-M-.

Hanshuai Cui, Zhiqing Tang, Zhi Yao, Weijia Jia, and Wei Zhao. Semantic-aware caching for efficient image generation in edge computing. *arXiv preprint arXiv:2511.22421*, 2025.

Chao Gong, Dong Li, Yingwei Pan, Jingjing Chen, Ting Yao, and Tao Mei. Freeinpaint: tuning-free prompt alignment and visual rationality enhancement in image inpainting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 4239–4247, 2026.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross-attention control. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=_CDixzkzeyb.

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 7514–7528, 2021.

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

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

Yi Huang, Jiancheng Huang, Yifan Liu, Mingfu Yan, Jiaxi Lv, Jianzhuang Liu, Wei Xiong, He Zhang, Liangliang Cao, and Shifeng Chen. Diffusion model-based image editing: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6):4409–4437, 2025.

Rosco Hunter, Łukasz Dudziak, Mohamed S Abdelfattah, Abhinav Mehrotra, Sourav Bhattacharya, and Hongkai Wen. Fast sampling through the reuse of attention maps in diffusion models. *International Journal of Computer Vision*, 133(9):6422–6431, 2025.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. volume 36, pp. 36652–36663, 2023.

Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132*, 2021.

Qinchan Li, Kenneth Chen, Changyue Su, Wittawat Jitkrittum, Qi Sun, and Patsorn Sangkloy. Cost-aware routing for efficient text-to-image generation. *arXiv preprint arXiv:2506.14753*, 2025.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Fei Ren, Shaobo Wang, Kaixin Li, and Linfeng Zhang. Speca: Accelerating diffusion transformers with speculative feature caching. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 10024–10033, 2025.

Chen-Yi Lu, Shubham Agarwal, Md Mehrab Tanjim, Kanak Mahadik, Anup Rao, Subrata Mitra, Shiv Kumar Saini, Saurabh Bagchi, and Somali Chaterji. Recon: Training-free acceleration for text-to-image synthesis with retrieval of concept prompt trajectories. In *European Conference on Computer Vision*, pp. 288–306. Springer, 2024.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11451–11461. IEEE, 2022.

Zhaoyang Lyu, Xudong Xu, Ceyuan Yang, Dahua Lin, and Bo Dai. Accelerating diffusion models via early stop of the diffusion process. *arXiv preprint arXiv:2205.12524*, 2022.

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *Advances in Neural Information Processing Systems*, 37: 133282–133304, 2024a.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15762–15772. IEEE, 2024b.

Hayk Manukyan, Andranik Sargsyan, Barsegh Atanyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Hd-painter: high-resolution and prompt-faithful text-guided image inpainting with diffusion models. In *International Conference on Learning Representations*, volume 2025, pp. 96301–96330, 2025.

Taehong Moon, Moonseok Choi, EungGu Yun, Jongmin Yoon, Gayoung Lee, and Juho Lee. Early exiting for accelerated inference in diffusion models. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling*, 2023.

Trong-Tung Nguyen, Quang Nguyen, Khoi Nguyen, Anh Tran, and Cuong Pham. Swiftedit: Lightning fast text-guided image editing via one-step diffusion. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21492–21501. IEEE, 2025.

OpenAI. DALL·E 2. `https://openai.com/dall-e-2`, 2023. Accessed: 2023.

Zizheng Pan, Bohan Zhuang, De-An Huang, Weili Nie, Zhiding Yu, Chaowei Xiao, Jianfei Cai, et al. T-stitch: Accelerating sampling in pre-trained diffusion models with trajectory stitching. In *International Conference on Learning Representations*, volume 2025, pp. 6103–6137, 2025.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Su Wang, Chitwan Saharia, Ceslee Montgomery, Jordi Pont-Tuset, Shai Noy, Stefano Pellegrini, Yasumasa Onoe, Sarah Laszlo, David J Fleet, Radu Soricut, et al. Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In *2023 ieee/cvf conference on computer vision and pattern recognition (cvpr)*, pp. 18359–18369. IEEE, 2023a.

Zijie J Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 893–911, 2023b.

Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al. Cache me if you can: Accelerating diffusion models through block caching. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6211–6220. IEEE, 2024.

Yuchen Xia, Divyam Sharma, Yichao Yuan, Souvik Kundu, and Nishil Talati. Modm: Efficient serving for image generation via mixture-of-diffusion models. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pp. 163–182, 2026.

Jiajian Xie, Shengyu Zhang, Zhou Zhao, Fan Wu, and Fei Wu. Ec-diff: Fast and high-quality edge-cloud collaborative inference for diffusion models. *arXiv preprint arXiv:2507.11980*, 2025.

Zewei Xin, Qinya Li, Chaoyue Niu, Fan Wu, and Guihai Chen. Adaptive routing of text-to-image generation requests between large cloud model and light-weight edge model. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19482–19491. IEEE, 2025.

Chenqian Yan, Songwei Liu, Hongjian Liu, Xurui Peng, Xiaojian Wang, Fangmin Chen, Lean Fu, and Xing Mei. Hybrid sd: Edge-cloud collaborative inference for stable diffusion models. *arXiv preprint arXiv:2408.06646*, 2024.

Hui Zhang, Zuxuan Wu, Zhen Xing, Jie Shao, and Yu-Gang Jiang. Adadiff: Adaptive step selection for fast diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 9914–9922, 2025.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. *arXiv preprint arXiv:2410.05317*, 2024.

A METHOD AND SYSTEM DETAILS

This section expands the method and system description of REMIX and makes the line of reasoning behind each design choice explicit. It first fixes the notation, then describes the serving pipeline and its data flow, then presents the pseudocode of the three procedures together with the motivation for each procedure, and finally analyzes the storage and computational overhead. The following section collects the additional experiments that support the claims of the main text.

A.1 NOTATION

Table 3 lists the symbols used in the main text and in this appendix. Steps are indexed so that x_0 is the clean latent and x_T is pure noise, and a larger step index therefore means a noisier latent. The two boundary steps $s_{\min}$ and $s_{\max}$ delimit the semantic planning stage of a generation, and the repainting start step s_{start} is always chosen inside this interval.

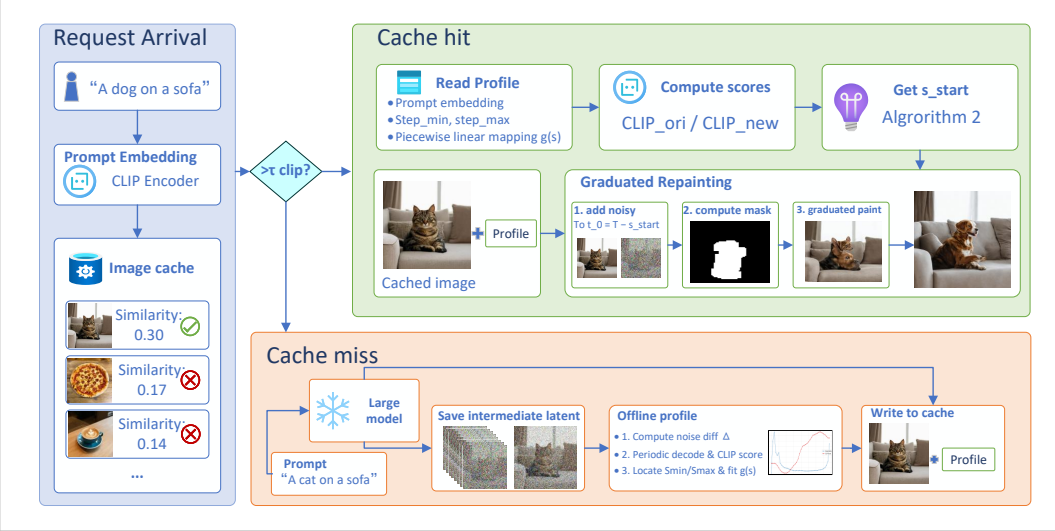


Figure 6: Overview of the REMIX serving pipeline. On a cache miss, the large model generates the image and profiles its denoising trajectory; on a cache hit, the lightweight model repaints the cached image using the stored profile.

A.2 SYSTEM OVERVIEW AND DATA FLOW

This section describes the end-to-end serving pipeline of REMIX and how data flows between its components. REMIX maintains a cache whose entries each store a generated image together with a compact profile, and it serves incoming requests with two diffusion models, a large model (SD3.5) that handles cache misses and a lightweight model (SD1.5) that handles cache hits.

Request arrival and cache lookup. When a request arrives, REMIX embeds its prompt and computes the CLIP score of each cached image under the incoming prompt. The request is a cache *hit* if some cached image attains a CLIP score above the threshold $\tau_{\text{clip}} = 0.26$, and a cache *miss* otherwise. A cache miss is served by the large model, and a cache hit is served by repainting the matched cached image with the lightweight model. Each cache entry holds the generated image, the source prompt it was generated from, and a compact profile of the generation trajectory.

Cache miss. The large model SD3.5 generates the image from scratch over T denoising steps. During generation, REMIX saves the intermediate latents and profiles the denoising trajectory in an offline thread. The thread computes the per-step noise-prediction delta Δ_i and periodically decodes the saved latents to obtain their CLIP alignment score. It then locates the boundary steps $s_{\min}$ and $s_{\max}$ by change-point detection, fits a monotone piecewise-linear step-to-CLIP mapping g, and writes the image, the source prompt, and this metadata back to the cache. The new entry

Symbol	Description
T	Total number of denoising steps of the large model (50 in our setting)
$\boldsymbol{\epsilon}_\theta$	Noise-prediction network
c, c_{orig}	Source prompt of a cached image
c_{new}	Prompt of an incoming request
x_i	Latent at denoising step i
Δ_i	Noise-prediction delta between steps $i-1$ and i
CLIP_i	CLIP alignment score of the latent at step i
$\text{SSE}(\cdot)$	Sum of squared residuals of the best linear fit to a sequence
$s_{\min}$	Minimum stable step, the start of the semantic planning stage
$s_{\max}$	Maximum effective step, the end of the semantic planning stage
g	Monotone piecewise-linear map from step to CLIP score
s_{start}	Repainting start step selected for a cache hit
τ_{clip}	CLIP threshold for a cache hit (0.26 in our setting)
$\text{CLIP}_{\text{orig}}$	Score of the cached image under its source prompt
CLIP_{new}	Score of the cached image under the new prompt
κ	Scaling factor that maps the semantic gap to a step offset
z_0	Clean latent of the cached image
$\boldsymbol{\eta}$	Shared noise tensor used for repainting
K	Number of repainting steps, equal to $T - s_{\text{start}}$
t_j	Timestep at repainting step j, with $t_j = K - j$
M_{diff}	Change map from the noise-prediction divergence
ρ_j	Graduated threshold at repainting step j
B_j	Binary repainting mask at repainting step j
z^{anchor}	Anchor state that preserves pixels outside the mask
$\bar{\alpha}_t$	Cumulative noise schedule coefficient at timestep t
$\mu(\cdot)$, $\sigma(\cdot)$	Spatial mean and standard deviation of a change map

Table 3: Symbols used in the main text and in this appendix.

becomes available to later requests as soon as the image is written. Because the profiling thread runs asynchronously, it does not delay the response.

Cache hit. The scheduler reads the profile of the matched entry, which stores the CLIP score of the cached image under its source prompt, $\text{CLIP}_{\text{orig}}$, and computes the score of the cached image under the new prompt, CLIP_{new}. From their gap it derives the repainting start step s_{start} through Algorithm 2. The lightweight model SD1.5 then performs Graduated Repainting. It noises the cached latent to the timestep $t_0 = T - s_{\text{start}}$, computes the change map and the graduated mask, and denoises only the changed regions while reproducing the rest through the shared-noise anchor state.

Data flow summary. The large model and the profiling thread run only on cache misses and write to the cache. The scheduler and the lightweight model run only on cache hits and read from it. Each cache entry stores the source prompt and a few trajectory statistics, adding only a few hundred bytes on top of the image, so the storage overhead is negligible. Figure 6 summarizes this flow.

A.3 Profiling-driven Adaptive Step Selection

Profiling-driven adaptive step selection consists of an offline stage and an online stage. Algorithm 1 runs offline during a cache miss and produces the profile metadata. Algorithm 2 runs online on a cache hit and maps the semantic gap to the repainting start step.

Profiling answers two questions that arise whenever a cached image has to be repainted. The first is where in the trajectory the repainting should begin, since an early start discards the layout the large model has already built while a late start cannot change the semantic content enough. The second is how strong the repainting should be for a given semantic gap. Both answers are contained in the trajectory itself. As discussed in the main text, the noise-prediction delta Δ_i and the CLIP alignment score CLIP_i divide a trajectory into three stages, namely structure forming, semantic planning, and fidelity improving. The repainting start step should lie inside the semantic planning stage, where the global layout is already fixed and the semantics are still fluid. Heuristic methods instead fix this step from a table indexed by prompt similarity, which ignores the trajectory and has to be re-tuned for every model and dataset.

Algorithm 1 Offline Profiling (on cache miss)

Require: large model ϵ_θ, source prompt c, total steps T, CLIP decoding stride m
Ensure: profile metadata $\big(c,\ \mathrm{CLIP}_{\mathrm{orig}},\ s_{\min},\ s_{\max},\ g(\cdot)\big)$

Generate the image from c, saving the latents $\{x_i\}_{i=0}^{T}$ and the noise predictions $\{\epsilon_\theta(x_i, i, c)\}_{i=1}^{T}$
for $i = 1, \ldots, T$ **do**
  $\Delta_i \leftarrow \|\epsilon_\theta(x_i, i, c) - \epsilon_\theta(x_{i-1}, i-1, c)\|_2$
end for
for $i \in \{m, 2m, \ldots, T\}$ **do**
  $\mathrm{CLIP}_i \leftarrow$ CLIP score of $\mathrm{decode}(x_i)$ under c
end for
$\mathrm{CLIP}_{\mathrm{orig}} \leftarrow$ CLIP score of the final decoded image under c
$s_{\min} \leftarrow \arg\min_s\ \mathrm{SSE}(\Delta_{1:s}) + \mathrm{SSE}(\Delta_{s+1:T})$
$s_{\max} \leftarrow \arg\min_{s \geq s_{\min}}\ \mathrm{SSE}(\mathrm{CLIP}_{s_{\min}:s}) + \mathrm{SSE}(\mathrm{CLIP}_{s+1:T})$
Fit a monotone piecewise-linear $g(s) = a_k s + b_k$ to the pairs $\{(s, \mathrm{CLIP}_s)\}$ over $[s_{\min}, s_{\max}]$, with $a_k \geq 0$

return $\big(c,\ \mathrm{CLIP}_{\mathrm{orig}},\ s_{\min},\ s_{\max},\ g(\cdot)\big)$

Change-point detection locates the two boundaries without introducing a threshold. For $s_{\min}$ we fit one straight line to $\Delta_{1:s}$ and another to $\Delta_{s+1:T}$, and we select the split that minimizes the total sum of squared residuals $\mathrm{SSE}(\Delta_{1:s}) + \mathrm{SSE}(\Delta_{s+1:T})$. The first segment follows the rapidly changing regime of structure forming, and the second follows the stable low-amplitude regime that comes after it. The same two-segment criterion applied to the CLIP curve over $[s_{\min}, T]$ gives $s_{\max}$, which separates the rising semantic planning stage from the saturating fidelity-improving stage. Restricting the search to $s \geq s_{\min}$ keeps the two boundaries ordered. The only free parameter in the procedure is the decoding stride m, so the boundaries adapt to each trajectory without manual calibration.

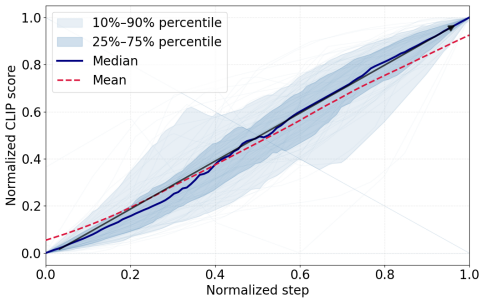


Figure 7: Normalized CLIP score versus normalized denoising step over $[s_{\min}, s_{\max}]$. Thin light-blue lines are individual trajectories, the dark-blue solid and red dashed lines are the median and the mean, and the shaded bands mark the 10–90% and 25–75% percentile ranges. The curves rise steadily, confirming monotonicity for the vast majority of samples.

Monotonicity of the CLIP trajectory. The monotone fit of Algorithm 1 assumes that the CLIP score rises over $[s_{\min}, s_{\max}]$, so that the mapping can be inverted. Figure 7 plots the normalized CLIP score against the normalized step for all profiled samples over $[s_{\min}, s_{\max}]$. Across COCO and DiffusionDB, the CLIP score is non-decreasing in 82% of all trajectories, and a further 17.4% contain fewer than three decreasing points, so 99.4% of trajectories are monotone or nearly monotone. The remaining 0.6% contain four or more decreasing points, which occur when $s_{\min}$ and $s_{\max}$ delimit an overly wide range that includes edge points at the two boundaries while the middle of the range remains monotone. This regularity is empirical rather than a theoretical guarantee, so the method does not rely on it for correctness. The fit g is computed under non-negative slope constraints ($a_k \geq 0$), which makes g monotone by construction and guarantees that the inverse g^{-1} is well-defined for every query.

The map g is fitted on the decoded pairs (s, CLIP_s) over $[s_{\min}, s_{\max}]$. We use a small number of segments so that the fit averages out decoding noise while preserving the overall shape of the curve, and the slopes are constrained to be non-negative so that g is non-decreasing by construction. The fitted map is stored as the segment breakpoints together with the slope and intercept of each segment, which occupies only a few dozen bytes.

Algorithm 2 Online Start-Step Query (on cache hit)

Require: cached metadata $\big(\mathrm{CLIP}_{\mathrm{orig}}, s_{\min}, s_{\max}, g(\cdot)\big)$, cached image, new prompt c_{new}, threshold τ_{clip}, total steps T
Ensure: start step s_{start}
$\mathrm{CLIP}_{\mathrm{new}} \leftarrow$ CLIP score of the cached image under c_{new}
if $\mathrm{CLIP}_{\mathrm{new}} \le \tau_{\text{clip}}$ **then**
  route the request to the cache-miss path and terminate
end if
$\mathrm{gap} \leftarrow \mathrm{CLIP}_{\mathrm{orig}} - \mathrm{CLIP}_{\mathrm{new}}$
$\kappa \leftarrow \dfrac{\mathrm{CLIP}_{\mathrm{orig}} - g(s_{\min})}{\mathrm{CLIP}_{\mathrm{orig}} - \tau_{\text{clip}}}$
$s_{\mathrm{start}} \leftarrow g^{-1}\big(\mathrm{CLIP}_{\mathrm{orig}} - \mathrm{gap} \cdot \kappa\big)$

return s_{start} {repainting strength $= (T - s_{\mathrm{start}})/T$}

The scaling factor κ anchors the two endpoints of the query. When the new prompt matches the cached image exactly, the gap is zero and $s_{\mathrm{start}} = g^{-1}(\mathrm{CLIP}_{\mathrm{orig}}) = s_{\max}$, so repainting uses the fewest steps. When the semantic gap is as large as the cache allows, $\mathrm{gap} = \mathrm{CLIP}_{\mathrm{orig}} - \tau_{\text{clip}}$ and $s_{\mathrm{start}} = g^{-1}(g(s_{\min})) = s_{\min}$, so repainting starts at the earliest effective step. Intermediate gaps interpolate between these two endpoints along the monotone fit g.

The offline stage adds one CLIP decode every m steps and a set of closed-form computations after generation. The delta terms reuse the noise predictions that were already computed during denoising, so the extra network cost is limited to the decodes. All of this runs after the response has been returned, in a separate thread, and the resulting metadata is amortized over every later hit on the same entry. Section B.3 measures the overhead.

A.4 Graduated Repainting

Graduated Repainting addresses the second limitation of full-image repainting. When the small model redraws every pixel, it regenerates regions that the large model already produced correctly, and the regenerated content drifts away from the cached image. The drift is most visible in the background, where the new prompt carries no information that constrains the redraw. Graduated Repainting therefore restricts the repaint to regions whose content has to change and keeps every other region identical to the cached image. Algorithm 3 describes the change-map computation, the graduated threshold, and the per-step mask-and-anchor blending.

The repainting schedule runs over $K = T - s_{\mathrm{start}}$ steps indexed by $j = 0, \ldots, K-1$, with timesteps $t_j = K - j$ decreasing from $t_0 = K$ to $t_K = 0$. The threshold ρ_j decays linearly from ρ_{init} to zero, so B_j expands monotonically from the core-change regions toward the whole image. The anchor state $z^{\mathrm{anchor}}_{t_{j+1}}$ is the cached clean latent z_0 noised to the current timestep with the same shared noise $\boldsymbol{\eta}$, so every pixel outside B_j is reproduced from the cached image instead of being re-synthesized. At the final step $\rho_{K-1} = 0$, so B_{K-1} covers the whole image, the anchor is fully released, and the output aligns with the new prompt.

Three design choices make the localization stable across steps. The initial threshold $\rho_{\mathrm{init}} = \mu(M_{\text{diff}}) + \sigma(M_{\text{diff}})$ scales to the statistics of each change map, so one rule covers both large and small edits. The 3×3 max pooling dilates the binarized region by one pixel and removes isolated pixels, which keeps the edited area connected. The anchor reuses the noise tensor $\boldsymbol{\eta}$ that produced z_{t_0}, so the anchored region follows the same noise trajectory as the initial noising and stays consistent with the cached image. Reusing the noise also avoids DDIM inversion, whose cost would dominate in a serving system.

A static binary mask leaves a hard boundary between the denoised region and the anchored region, and this discontinuity appears as a visible seam. The graduated threshold removes the seam by sweeping the boundary outward over the K steps. Each region joins the denoised set at its own step rather than all at once, so no pixel jumps from the anchored state to the denoised state in a single step. Because the two states share the same noise level at every step, the boundary between

Algorithm 3 Graduated Repainting (on cache hit)

Require: cached clean latent z_0, source prompt c_{orig}, new prompt c_{new}, start step s_{start}, total steps T, shared noise $\boldsymbol{\eta}$
Ensure: repainted latent
$K \leftarrow T - s_{\text{start}}$ {number of repainting steps}
$t_0 \leftarrow K$; $\quad z_{t_0} \leftarrow \sqrt{\bar{\alpha}_{t_0}}\, z_0 + \sqrt{1-\bar{\alpha}_{t_0}}\, \boldsymbol{\eta}$
$\boldsymbol{\epsilon}_{\text{orig}} \leftarrow \boldsymbol{\epsilon}_\theta(z_{t_0}, t_0, c_{\text{orig}})$; $\quad \boldsymbol{\epsilon}_{\text{new}} \leftarrow \boldsymbol{\epsilon}_\theta(z_{t_0}, t_0, c_{\text{new}})$
$M_{\text{diff}} \leftarrow \|\boldsymbol{\epsilon}_{\text{new}} - \boldsymbol{\epsilon}_{\text{orig}}\|_2$, normalized to $[0,1]$ and smoothed by max pooling
$\rho_{\text{init}} \leftarrow \mu(M_{\text{diff}}) + \sigma(M_{\text{diff}})$
for $j = 0, \ldots, K-1$ **do**
  $t_{j+1} \leftarrow t_j - 1$
  $\rho_j \leftarrow \rho_{\text{init}} \cdot \big(1 - \frac{j}{K-1}\big)$
  $B_j \leftarrow \text{MaxPool}_{3\times 3}\big(\mathbb{1}[M_{\text{diff}} > \rho_j]\big)$
  $z_{t_{j+1}}^{\text{denoised}} \leftarrow \text{Denoise}(z_{t_j}, t_j, B_j, c_{\text{new}})$ {classifier-free guidance inside B_j}
  $z_{t_{j+1}}^{\text{anchor}} \leftarrow \sqrt{\bar{\alpha}_{t_{j+1}}}\, z_0 + \sqrt{1-\bar{\alpha}_{t_{j+1}}}\, \boldsymbol{\eta}$
  $z_{t_{j+1}} \leftarrow B_j \odot z_{t_{j+1}}^{\text{denoised}} + (1 - B_j) \odot z_{t_{j+1}}^{\text{anchor}}$
end for

return decode(z_{t_K}), with $t_K = 0$

Method	Cached content	Cache size per entry
SD3.5 (upper bound)	none (generates from scratch)	0
CacheGenius	reference image + semantic features	1 image + features
MoDM	final generated image + source prompt	1 image + prompt
DiffEdit	none (edits from scratch)	0
Nirvana	latent at all T steps + prompt embedding	$T\times$ latent + prompt
REMIX (ours)	final image + source prompt + profile metadata	1 image + $O(1)$ bytes

Table 4: Storage composition and per-entry cache size of REMIX and the baselines.

them differs only in content, and the max pooling softens the transition further. At the last step the threshold reaches zero, the mask covers the whole image, and the anchor is fully released, so the final image contains no anchored pixels. A fixed binarization threshold, as used by mask-based editing methods, does not have this property and is therefore subject to the seam. Section B.1 examines the mask response on foreground and background regions.

A.5 STORAGE AND COMPLEXITY ANALYSIS

This section analyzes the storage and computational overhead of REMIX relative to the baselines. Table 4 lists the cache composition and per-entry size of each method. REMIX stores the final image plus a compact profile (the source prompt, $\text{CLIP}_{\text{orig}}$, $s_{\min}$, $s_{\max}$, and the parameters of the piecewise-linear g), which amounts to a few hundred bytes and is negligible compared with the image. By contrast, Nirvana caches the latent at every denoising step together with the prompt, so its per-entry storage scales linearly with T. Profiling runs offline and asynchronously and is amortized across the serving lifetime. Repainting runs at most $T - s_{\text{start}} \le T$ steps, and the anchor region is produced by closed-form noising rather than by extra network passes.

B ADDITIONAL EXPERIMENTS

This section collects the additional experiments that support the main claims of the text and, in each case, traces the measured advantage back to the specific design choice that produces it. Section B.1 verifies that the change map localizes the region that must change. Section B.2 shows that the same behavior holds across four different types of prompt change. Section B.3 measures the cost of profiling. All experiments follow the setup of the main text, with SD3.5 Medium as the large model, SD1.5 as the small model, 512×512 resolution, and $T = 50$ denoising steps, and all comparisons

within a table or figure use the same set of cache-edit pairs, so a difference between two rows or two panels is attributable to the factor under study.

B.1 Graduated Mask Analysis on Foreground and Background

Setting and purpose. Graduated Repainting rests on the assumption that the change map M_{diff} activates the region that must change and stays quiet on the region that should be preserved, because a mask that fails this assumption would either under-edit the request or damage content that the request never mentioned. This experiment tests the assumption directly. We build two groups of cache-edit pairs. In the foreground group the object changes while the scene is held fixed, for example "a cat on a sofa" to "a dog on a sofa". In the background group the scene changes while the object is held fixed, for example "a cat on a sofa" to "a cat on a bed". For every pair we noise the cached latent to t_0, compute M_{diff} from the two noise predictions, and overlay the response on the cached image.

Result. Figure 8 shows the response for a representative pair. The activation concentrates on the object in the foreground group and on the scene in the background group, which is exactly the region the two prompts disagree about, while the region the two prompts describe identically remains weakly activated. The localization is therefore correct in both directions, and it does not depend on which part of the image is being edited.

Why the map localizes. The response is a by-product of the model's own semantic comparison rather than the output of a dedicated module. Both noise predictions are evaluated at the same noisy latent z_{t_0}, so wherever the two prompts assert the same content the two predictions nearly coincide and their difference vanishes, and the difference grows only where the prompts assert different content. No segmentation network, attention map, or ground-truth mask is involved. The raw map is not yet usable as a mask on its own, since it is noisy and both over- and under-inclusive near the boundaries, and this is precisely why the threshold is graduated over the repainting steps instead of being applied once.

B.2 Qualitative Results across Prompt Categories

A serving method is useful only if it handles the kinds of prompt changes that real requests contain, not a single edit type. We therefore group edits by what the new prompt asks for, namely object identity ("a cat on a sofa" to "a dog on a sofa"), attribute ("a red car" to "a blue car"), background ("a cat on a sofa" to "a cat on a bed"), and spatial composition, and we show one representative pair per category in Figures 9–12. Each pair is served by ReMix, by MoDM, and by the large model SD3.5 generating from scratch.

In the identity and attribute categories, ReMix replaces the object or the attribute and leaves the rest of the scene untouched, whereas MoDM's full-image repainting drifts over the whole image and modifies content that the request never mentioned. In the background category the roles reverse, and ReMix repaints the scene while preserving the object. In the composition category, ReMix rearranges the layout while keeping the appearance of the objects themselves. SD3.5 produces a valid image in every case, but because it never reads the cached image it also discards the parts of the scene that the request did not ask to change. The behavior of ReMix follows the request in all four categories rather than being tuned to any one of them.

Two mechanisms act on different axes, and neither is specific to a category. The adaptive start step s_{start} sets how much of the trajectory is redrawn: a small semantic gap places s_{start} near $s_{\max}$ and redraws only the later, detail-bearing steps, which keeps the coarse structure of the cached image, while a large gap moves s_{start} toward $s_{\min}$. The graduated mask then sets where the redraw is allowed to land. Because the extent of the edit is decided by the measured gap and its location is decided by the measured change map, both adapt per request, so the same two mechanisms cover an identity edit and a background edit without any category-specific setting.

B.3 Profiling Overhead

Profiling decomposes into a synchronous part that records intermediate state during generation and an offline part that analyzes the recorded trajectory. We measure both parts against a plain SD3.5

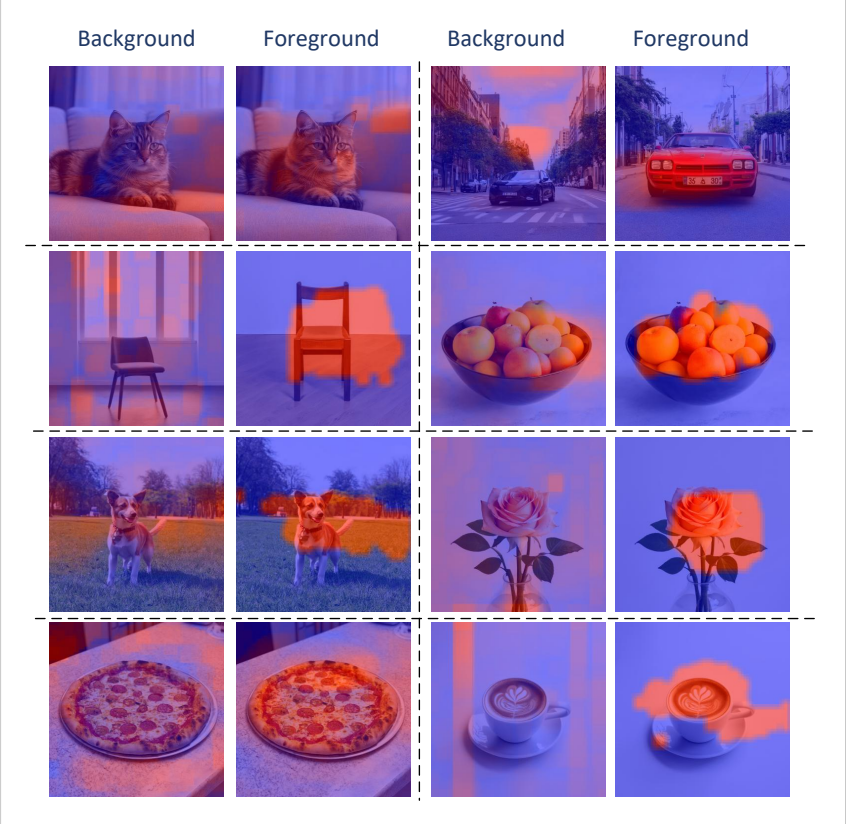


Figure 8: Foreground and background recognition heatmap of the graduated mask. Strong responses concentrate on the foreground regions that must change, while the background regions that should be preserved remain weakly activated.

Component	Time (s)	Fraction of one generation
SD3.5 generation	8.610	100%
Latent recording and delta computation	0.416	4.8%
Periodic CLIP decoding and scoring	9.006	104.6%
Total profiling overhead	9.422	109.4%

Table 5: Profiling overhead relative to a single SD3.5 generation. Only the latent recording lies on the critical path. The CLIP decoding runs in an offline thread and does not delay the response.

generation on the same device. A plain generation takes 8.610 s. Recording the intermediate latents together with the per-step noise deltas adds 0.416 s on the critical path, which is 4.8% of a generation. The offline thread then decodes the saved latents at stride m and scores each decoded image with CLIP, which takes 9.006 s, or 104.6% of a generation. Table 5 reports the breakdown.

Two observations follow. First, the offline analysis dominates the profiling cost and is itself dominated by the periodic CLIP decoding rather than by the delta computation, since the deltas reuse the noise predictions that the generation already produced. Second, only the 0.416 s recording lies on the critical path, and this is the quantity that enters the latency of a cache miss; the 9.006 s analysis runs in a separate thread after the response has been returned. Because the metadata is written once per cache entry, the analysis cost is paid a single time and is amortized across all later hits on that entry. Profiling one image end to end therefore takes about 18.0 s in total, of which 8.6 s is the generation itself. The two change-point searches and the monotone fit operate in closed form on the 50-step sequences, and their cost is negligible compared with decoding, so they are not timed separately here.

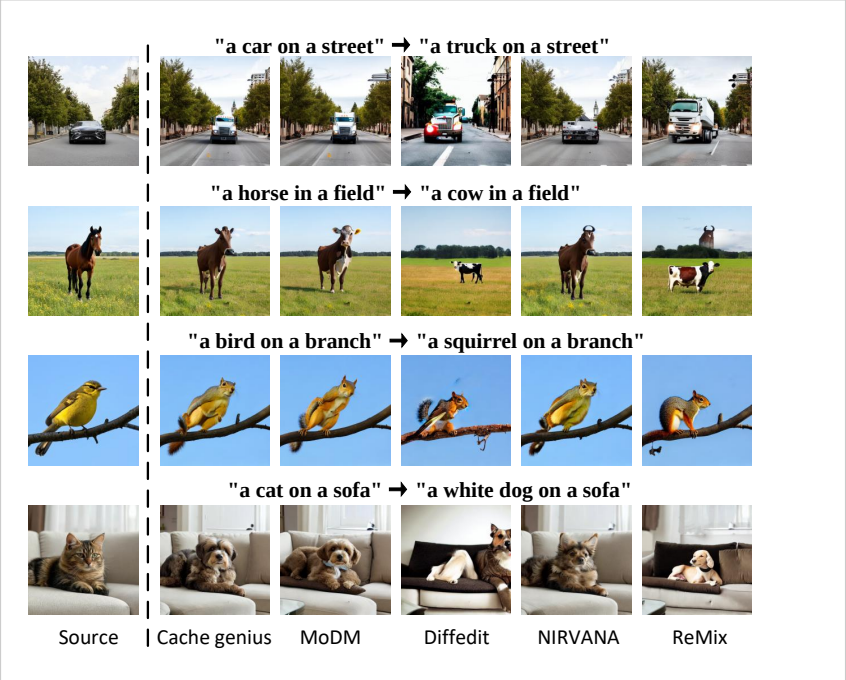


Figure 9: Object identity change. REMIX repaints only the object while preserving the background, whereas MoDM's full-image repainting alters the whole scene.

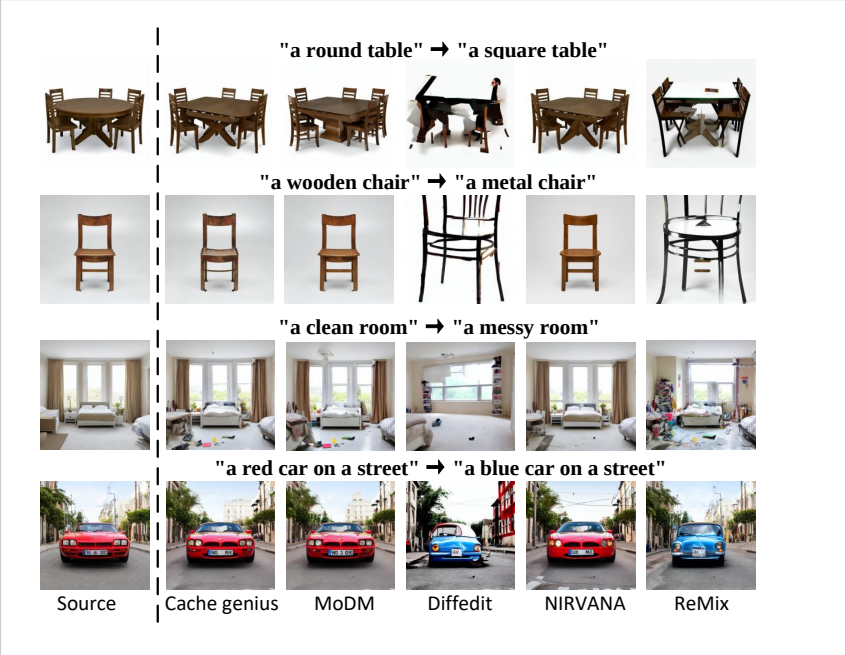


Figure 10: Attribute change. REMIX edits only the changed attribute while keeping the object and background faithful to the cached image.



Figure 11: Background change. REMIX repaints the background while keeping the foreground object faithful to the cached image.

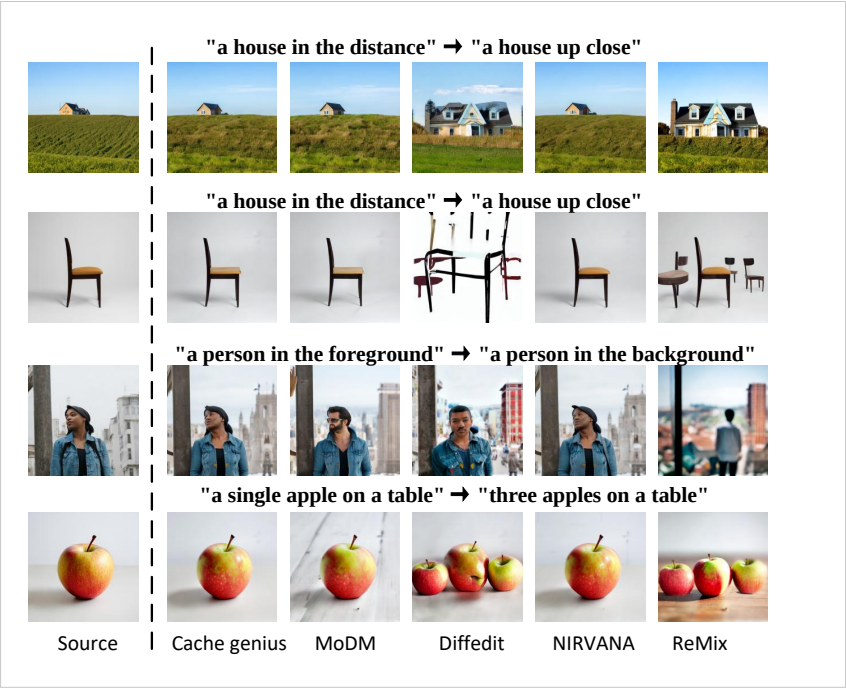


Figure 12: Spatial composition change. REMIX adjusts the layout while preserving the appearance of the objects.