LIGHTDRAFT: ACCELERATING MASKED DIFFUSION IMAGE EDITING WITH LOOK-AHEAD GUIDANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Masked diffusion models support instruction-guided image editing through parallel token prediction. However, their reliance on bidirectional attention requires repeated full-sequence computation throughout sampling, while a fixed sampling schedule assigns the same refinement budget to edits with very different spatial extent and convergence behavior. To address this problem, we investigate whether intermediate predictions can reveal where computation is needed and how long refinement should continue. We observe that the spatial support of an edit often stabilizes before its visual appearance is fully refined. Based on this observation, we propose **LightDraft**, a training-free acceleration method that adapts both feature computation and sampling length to each edit. LightDraft decodes intermediate complete token predictions into **look-ahead images**, exposing the developing edit before sampling is complete. Differences from the reference image are used to progressively localize the edit region and guide a two-stage regional caching strategy, allowing most features outside the region to be reused. Meanwhile, changes between successive look-ahead images guide the sampling budget, reducing unnecessary refinement when predictions have largely stabilized. Experiments on ImgEdit and PIE-Bench show that LightDraft achieves an **8.14× speedup** while maintaining instruction-following capability and editing quality.

1 INTRODUCTION

Diffusion large language models (dLLMs) have brought masked-token generation to a growing range of language and vision tasks (Nie et al., 2025; Li et al., 2026a; Xin et al., 2025). For image editing, models such as LaViDa-O and Lumina-DiMOO generate an edited image from a reference image and a textual instruction. This formulation supports parallel decoding, but its computational cost remains high. In the standard Lumina-DiMOO editor, every sampling step applies bidirectional attention to the entire sequence. Even an edit confined to one object repeatedly processes the surrounding scene, although most of that scene should remain unchanged. The resulting cost depends on both how many positions are recomputed at each step and how many refinement steps are performed. Efficient editing therefore calls for decisions about where computation is needed and how long refinement should continue.

Recent work has accelerated image and text generation through feature caching and improved sampling (Ma et al., 2024; Liu et al., 2025b; Wu et al., 2025; Wei et al., 2025). Editing methods such as TurboEdit (Deutch et al., 2024) and FireFlow (Deng et al., 2025) reduce the cost of inversion in continuous diffusion and rectified flow models. Their inversion procedures, however, do not directly apply to Lumina-DiMOO, which generates discrete visual tokens from a fully masked target sequence rather than inverting the reference image into noise. Sparse-LaViDa (Li et al., 2026b) accelerates multimodal sampling through sparse modeling and additional training. Here, we focus on exploiting the locality of an edit without changing the pretrained model. The challenge is to identify the edit region early enough to avoid spending most of the sampling budget on full-image computation.

To understand when the edit region becomes apparent, we construct look-ahead images from intermediate token predictions. At each step, the model predicts tokens at all remaining masked positions, while the sampler keeps only a subset of the newly predicted tokens. We decode the complete

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: **From region localization to detail refinement.** Look-ahead images (top) and difference regions relative to the reference image (bottom, purple) reveal how edits evolve during sampling. Early regions change in location and extent, while later regions become more spatially stable as appearance continues to develop. This motivates combining early candidate regions and updating the region for refinement.

token prediction, which combines predictions at masked positions with previously unmasked tokens, without changing the sampling state.

Because each look-ahead image contains a prediction at every spatial position, it can be compared with both the reconstructed reference image and predictions from other steps. These comparisons provide a common image-space signal for locating the edit and tracking its evolution.

Figure 1 illustrates a transition from region localization to detail refinement. Early in sampling, differences from the reconstructed reference image change in location and extent, and some initial detections disappear in later predictions. As sampling proceeds, the estimated edit region becomes more spatially stable, while local structure and texture continue to develop. This suggests that *the edit region can stabilize before its appearance does*.

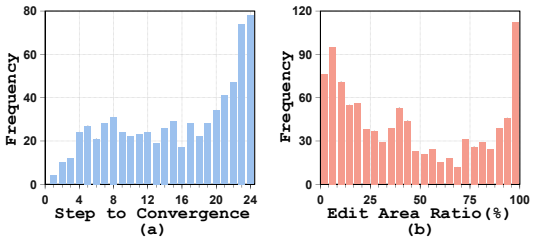


Figure 2: **Illustrative variation in editing requirements.** Panels illustrate variation in (a) refinement duration and (b) spatial extent of editing. The distributions are schematic and should not be interpreted as empirical measurements.

This observation motivates **LightDraft**, a training-free method that adapts feature caching and sampling length to each edit. The central challenge is selecting where to cache, since edit location and extent vary across instructions and evolve during sampling. We compare look-ahead images with the reconstructed reference to guide two-stage regional caching. During localization, early candidate regions are combined; features inside their union are recomputed and most features outside it are reused. *Combining early regions avoids reliance on a single uncertain prediction.* At a predefined update step, the current look-ahead image provides a refined region for later caching, removing early detections whose differences no longer exceed the threshold. The two stages balance early coverage with later feature reuse.

Figure 2 schematically illustrates two dimensions of an edit's computational requirements: refinement duration and spatial extent. These dimensions motivate adapting computation to each edit.

Our adaptive sampling budget compares recent changes between successive look-ahead images with changes during initialization. We aggregate these changes inside and outside the estimated region, accounting for their amount and spatial extent. Smaller relative changes favor a smaller budget, while larger changes allocate more refinement steps. Continued changes in either the edited region or the background can therefore influence the allocation. The budget is set once at the predefined

region update step t_1. Thus, *reference differences guide where to compute, while changes between look-ahead images guide how long to refine each edit*.

Our contributions are summarized as follows:

- **Look-ahead images for early edit localization.** We introduce look-ahead images that expose the developing edit before token sampling is complete. We decode the complete prediction formed from newly predicted tokens and previously unmasked tokens, while leaving the sampling state unchanged. Comparing these images with the reconstructed reference image provides spatial evidence for early edit localization, and comparing successive images reveals how the prediction continues to change.
- **Two-stage regional feature caching.** We introduce a two-stage caching strategy that updates the estimated edit region during sampling. Candidate regions from early look-ahead images are first aggregated to capture uncertain edit locations. The support is then updated at a predefined step, allowing computation to focus on the refined edit region while most features elsewhere are reused.
- **Adaptive sampling budget.** We introduce an adaptive budget that compares recent changes between look-ahead images with those observed during initialization. The allocation accounts for both the amount and spatial extent of change, inside and outside the estimated edit region. The total budget is set once at the region update, reusing the same decoded predictions. Together with regional caching, this adapts both per-step computation and sampling length to each edit.

2 Related Work

Masked diffusion models. Masked modeling predicts missing content from visible context. BERT and BEiT (Devlin et al., 2019; Bao et al., 2021) established masked pretraining for language and visual representation learning, respectively. MaskGIT (Chang et al., 2022) introduced confidence-based parallel prediction of discrete visual tokens, and Muse (Chang et al., 2023) extended this approach to text-conditioned synthesis. More recently, MMaDA (Yang et al., 2026) unified multimodal understanding and generation, while Lumina-DiMOO and LaViDa-O (Xin et al., 2025; Li et al., 2026a) enabled instruction-guided editing conditioned on a reference image. Parallel prediction reduces sequential token updates, but bidirectional attention still requires repeated context-dependent computation.

Image editing. Early generative editing explored GAN latent spaces (Shen et al., 2020); diffusion models subsequently broadened the supported tasks (Huang et al., 2025). SDEdit (Meng et al., 2021) perturbs the reference image with noise and recovers an edited image through denoising. Prompt-to-Prompt (Hertz et al., 2022) controls cross-attention to preserve spatial structure, while Null-text Inversion (Mokady et al., 2023) optimizes unconditional embeddings around an inversion trajectory. DiffEdit (Couairon et al., 2022) estimates editing masks from differences between text-conditioned predictions. InstructPix2Pix (Brooks et al., 2023) learns instruction following from paired images and instructions. FLUX.1 Kontext and Mage-Flow-Edit (Labs et al., 2025; Zhang et al., 2026) support instruction-guided editing through flow-matching-based generation. These methods control edits through attention, inversion, masks, or learned conditioning. Our work addresses inference efficiency in masked diffusion editors.

3 Method

LightDraft uses look-ahead images to localize edits and allocate computation. Differences from the reconstructed reference guide two-stage regional caching, while changes between successive predictions determine the sampling budget (Figure 3).

3.1 Preliminaries

Masked diffusion image editing. Given a reference image X_{ref} in pixel space and a textual instruction c, masked diffusion models (Li et al., 2026a; Xin et al., 2025) generate the edited image

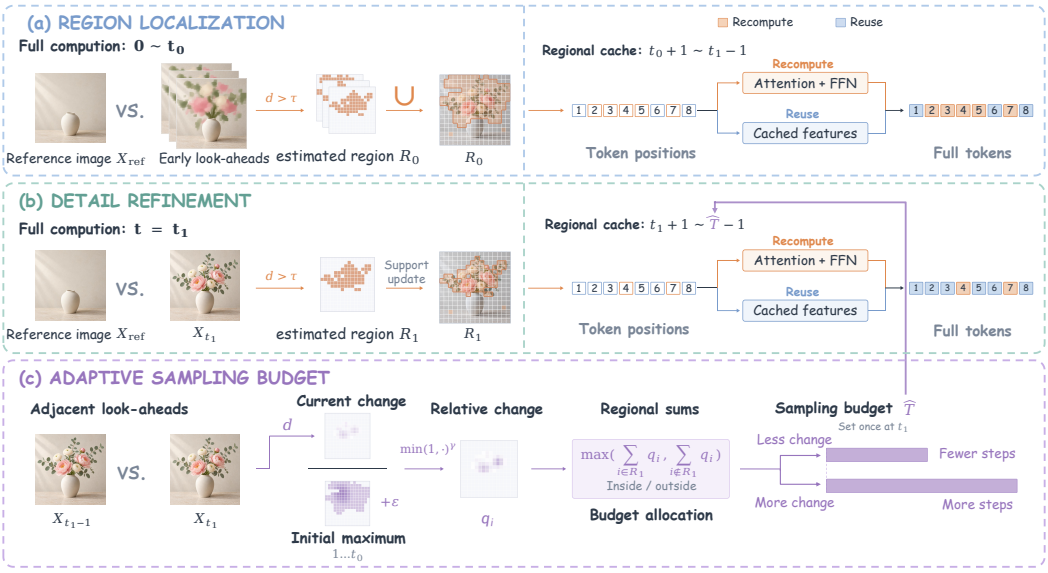


Figure 3: **Overview of LightDraft.** (a) Early look-ahead images are compared with the reconstructed reference to form a union of candidate edit regions for regional caching. (b) At t_1, a full forward pass updates the region and its computational support for detail refinement. Orange and blue denote recomputed and reused features, respectively. (c) Recent prediction changes, normalized by their initialization scale, are aggregated inside and outside the estimated edit region to set the total sampling budget $\widehat{T}$ once at t_1.

through iterative unmasking. The reference image X_{ref} and textual instruction c are encoded as conditioning prefix tokens, while the target image is represented by n visual tokens initialized entirely as $\mathbf{x}_0 = [\texttt{MASK}]^n$. At each step, the model predicts tokens at all currently masked positions and uses prediction confidence to decide which positions to unmask. Once a token is unmasked, it remains fixed in all subsequent steps. Repeating this process gradually completes the image.

More concretely, at step t, let $\mathcal{U}_t = \{i \in \Omega : x_{t,i} = \texttt{MASK}\}$ denote the masked positions, where $\Omega = \{1, \ldots, n\}$. A bidirectional Transformer predicts tokens at these positions in parallel:

$$(\hat{\mathbf{x}}_t, \mathbf{s}_t) = \text{Predict}_\theta(\mathbf{x}_t; X_{\text{ref}}, c), \tag{1}$$

where θ denotes the pretrained model parameters. Here, $\hat{\mathbf{x}}_t$ is the complete token prediction at step t: it contains the newly predicted tokens at positions in $\mathcal{U}_t$ and keeps all previously unmasked tokens unchanged. The vector $\mathbf{s}_t$ records the confidence scores of the newly predicted tokens.

The sampler then uses these confidence scores to decide which newly predicted tokens to keep and which ones to mask again:

$$\mathbf{x}_{t+1} = \text{Remask}_t(\hat{\mathbf{x}}_t; \mathbf{s}_t, \mathcal{U}_t). \tag{2}$$

Only positions in $\mathcal{U}_t$ can be remasked; tokens that were already unmasked before step t remain unchanged. Thus, $\hat{\mathbf{x}}_t$ is the complete token prediction at the current step, while $\mathbf{x}_{t+1}$ is the partially masked token sequence used for the next step. We follow the base model's confidence-based sampling procedure with random perturbations and a cosine masking schedule that gradually removes all masks over the allotted sampling steps. Let T denote the nominal sampling budget, with steps indexed from zero.

Naïve feature caching. Predicting many tokens in parallel still requires computing features for all input positions at every step. Feature caching reduces this repeated work by reusing earlier representations (Liu et al., 2025b; Wu et al., 2025). A simple policy reuses cached features at a fixed fraction of token positions between full forward passes. At a fixed layer and attention head, let $\mathcal{A}_t$ denote the positions in the full input sequence whose features are recomputed. The attention

mapping $\mathcal{F}_{\text{attn}}$ computes their outputs using keys and values from all positions:

$$\mathcal{F}_{\text{attn}}\big(\mathbf{Q}_{t,\mathcal{A}_t}, \bar{\mathbf{K}}_t, \bar{\mathbf{V}}_t\big) = \text{softmax}\left(\frac{\mathbf{Q}_{t,\mathcal{A}_t}\bar{\mathbf{K}}_t^\top}{\sqrt{d_h}}\right)\bar{\mathbf{V}}_t. \tag{3}$$

Here $\mathbf{Q}_{t,\mathcal{A}_t}$ contains queries at the recomputed positions. The matrices $\bar{\mathbf{K}}_t$ and $\bar{\mathbf{V}}_t$ combine new and cached entries over the full sequence, including the conditioning prefix, and d_h is the attention-head dimension. Positions using cached features skip attention and feed-forward computation. This is an approximation: features can change through bidirectional attention even when their input tokens remain fixed. Moreover, a fixed caching ratio does not account for the location or size of the edit.

3.2 Look-Ahead Images for Edit Localization

The partially masked token sequence $\mathbf{x}_{t+1}$ provides an incomplete view of the edit during early sampling, since many positions still contain `MASK` tokens with no visual content. However, $\hat{\mathbf{x}}_t$ already contains predictions at all these masked positions, together with previously unmasked tokens. We therefore use the pretrained image decoder $\mathcal{D}$ to obtain the **look-ahead image** $X_t = \mathcal{D}(\hat{\mathbf{x}}_t)$ at selected steps, before remasking. Unlike the partially masked token sequence $\mathbf{x}_{t+1}$, this image shows the predicted content at every position and thus provides an early estimate of the final edited image.

Locating the edit early. We use differences from the reference image to identify where features should be recomputed. First, we obtain the reconstructed reference image $X_{\text{rec}} = \mathcal{D}(\mathcal{E}(X_{\text{ref}}))$, where $\mathcal{E}$ is the image tokenizer. Both X_{rec} and the look-ahead images are produced by the same decoder. For any two aligned pixel-space images X and Y, a difference operator $\mathcal{F}_{\text{diff}}$ produces a nonnegative map on the visual-token grid:

$$\mathbf{d}(X, Y) = \mathcal{F}_{\text{diff}}(X, Y) \in \mathbb{R}^n_{\geq 0}. \tag{4}$$

We use $\mathcal{F}_{\text{diff}}(X, Y) = \frac{1}{2}\mathcal{P}\big(\mathbf{1} - \text{SSIM}_{\text{RGB}}(X, Y)\big)$, where SSIM_{RGB} is the channel-averaged local SSIM map (Wang et al., 2004), $\mathbf{1}$ is the all-ones map, and $\mathcal{P}$ averages pixels within each visual token. The entry $d_i(X, Y)$ measures the local difference at token position i, with larger values indicating greater differences. Locations above a fixed threshold τ define the estimated edit region. This region guides feature recomputation; it does not prevent changes elsewhere in the output.

3.3 Two-Stage Regional Caching

Figure 1 motivates two caching stages, illustrated in Figure 3(a,b). During **region localization**, candidate edit regions can change in location and extent. We begin with full forward passes to collect these regions, then **recompute features throughout their union** while reusing most features outside it. During **detail refinement**, the estimated edit region tends to become spatially stable. We therefore update the region and **concentrate feature recomputation within it**, while reusing most features outside the estimated edit region.

Stage I: region localization. Early in region localization, candidate edit regions can shift and change in extent, so a single look-ahead image may miss potential edit locations. We therefore start with full forward passes and combine the candidate edit regions detected at each step. Let t_0 mark the end of initialization and t_1 the later region update step, with $1 \leq t_0 < t_1 < T$. Both steps are fixed in advance. The initial look-ahead images define

$$\mathcal{R}_0 = \bigcup_{t=0}^{t_0} \{i \in \Omega : d_i(X_{\text{rec}}, X_t) > \tau\}. \tag{5}$$

This union defines the candidate edit region $\mathcal{R}_0$. From step $t_0 + 1$ to $t_1 - 1$, we recompute features within this region and reuse most features outside it.

Stage II: detail refinement. At the transition to detail refinement, $\mathcal{R}_0$ may still include areas whose differences from the reconstructed reference image no longer exceed the threshold. We therefore perform a full forward pass and update the estimated edit region at step t_1:

$$\mathcal{R}_1 = \{i \in \Omega : d_i(X_{\text{rec}}, X_{t_1}) > \tau\}. \tag{6}$$

Let α denote the minimum computational support size as a fraction of the n target-image tokens ($\alpha = 0.10$ by default). If $|\mathcal{R}_1| < \lceil \alpha n \rceil$, we expand it to $\widetilde{\mathcal{R}}_1$ by adding outside positions in descending order of their difference scores $d_i(X_{\text{rec}}, X_{t_1})$, until it contains $\lceil \alpha n \rceil$ tokens. Otherwise, we set $\widetilde{\mathcal{R}}_1 = \mathcal{R}_1$. We use the original thresholded region $\mathcal{R}_1$ to allocate the sampling budget, while the computational support $\widetilde{\mathcal{R}}_1$ is used for feature caching after t_1 and for output compensation. At each step, $\mathcal{A}_t$ additionally includes background positions selected for refresh and positions whose input tokens have changed.

In both stages, features inside the current region are recomputed. Outside the region, the caching ratio ρ specifies the fraction of target-image tokens that reuse cached features, while the remaining tokens are recomputed.

Region-guided residual correction (RRC). At the final sampling step t, we reuse the expanded support $\widetilde{\mathcal{R}}_1$ of the estimated edit region to guide output correction. The reference residual $X_{\text{ref}} - X_{\text{rec}}$ compensates reconstruction differences outside this support; applying it within edited regions could reintroduce reference details that no longer match the generated content.

Let $\mathbf{m}$ be the support's binary mask bilinearly upsampled to the image grid. For aligned RGB images in $[0, 1]$, we apply the same mask to all channels and form the output by projecting $X_t + (\mathbf{1} - \mathbf{m}) \odot (X_{\text{ref}} - X_{\text{rec}})$ pointwise onto $[0, 1]$, where $\odot$ denotes elementwise multiplication. Before projection, locations where the mask is zero retain the generated change $X_t - X_{\text{rec}}$ on top of the reference image. This correction changes only the decoded output, leaving token sampling unchanged.

3.4 Adaptive Sampling Budget

Different edits vary in spatial extent and in how long their predictions keep changing. Using the same number of sampling steps for every edit ignores these differences. As illustrated in Figure 3(c), we adapt the total sampling budget to changes between successive look-ahead images: smaller changes relative to initialization lead to a smaller budget.

Measuring relative change. Let $b_{t,i} = d_i(X_{t-1}, X_t)$ denote the change at token position i between successive look-ahead images, for $t \geq 1$. At the region update step t_1, we compare the latest change with the largest change observed during initialization:

$$q_i = \min\left(1, \frac{b_{t_1,i}}{\max_{1 \leq r \leq t_0} b_{r,i} + \epsilon}\right)^{\gamma}. \tag{7}$$

Here $\epsilon > 0$ is a small constant and $\gamma \geq 1$ controls sensitivity to a decline in change. A smaller q_i indicates less recent change relative to initialization. This score measures prediction change, not instruction fulfillment, and may include the effect of resuming full computation at t_1.

Allocating the total sampling budget. Let $T_{\min}$ and $T_{\max}$ be the minimum and maximum total budgets, with $t_1 < T_{\min} \leq T_{\max} \leq T$. For a nonempty thresholded region $\mathcal{R}_1$, we sum the scores inside and outside the region and use the larger contribution to allocate the total number of sampling steps:

$$\widehat{T} = \min\left(T_{\max}, \left\lfloor T_{\min} + \frac{T - T_{\min}}{n} \max\left(\sum_{i \in \mathcal{R}_1} q_i, \sum_{i \in \Omega \setminus \mathcal{R}_1} q_i\right)\right\rfloor\right). \tag{8}$$

Here $\lfloor \cdot \rfloor$ denotes the floor function. Normalizing both regional sums by the full grid size n accounts for the amount and area of change, and taking the larger contribution allows continued background changes to increase the budget. Smaller relative changes favor $T_{\min}$, while larger changes increase the budget up to $T_{\max}$. If $\mathcal{R}_1$ is empty, we set $\widehat{T} = T_{\max}$, irrespective of subsequent support expansion. Unless otherwise specified, we use $T = 24$, $T_{\min} = 12$, and $T_{\max} = 18$.

The budget is set once, before remasking at t_1, leaving $\widehat{T} - t_1 - 1$ refinement steps. Step allocation uses the initial look-ahead images and the pair (X_{t_1-1}, X_{t_1}), reusing images already available from region estimation. All image decoding and comparison costs are included in end-to-end latency.

4 EXPERIMENTS

We evaluate editing quality, background preservation, and inference efficiency on ImgEdit and PIE-Bench. We compare LightDraft with the 64-step baseline and existing acceleration methods, then analyze the contributions of regional processing and adaptive sampling.

4.1 EXPERIMENTAL SETUP

Model and benchmarks. We evaluate our method using the pretrained Lumina-DiMOO model (Xin et al., 2025) without additional training. On ImgEdit (Ye et al., 2026), we evaluate 637 single-turn, non-style instructions across 166 source images and eight editing categories. PIE-Bench (Ju et al., 2023) contains 700 editing tasks. Background metrics are averaged over the 556 tasks with a nonempty unedited region; all 700 tasks contribute to CLIP-Edit and latency measurements. Images follow the model's aspect-ratio-aware preprocessing, and annotated regions are used only for evaluation.

Default settings. Unless otherwise specified, our method uses $t_0 = 2$ and $t_1 = 8$ under zero-based indexing, corresponding to the third and ninth iterations, respectively. We set the region threshold to $\tau = 0.04$, the minimum support fraction to $\alpha = 0.10$, and the background caching ratio to $\rho = 0.95$. For adaptive sampling, we use the nominal budget $T = 24$, bounds $T_{\min} = 12$ and $T_{\max} = 18$, exponent $\gamma = 2$, and numerical stability constant $\epsilon = 10^{-8}$.

Baselines and metrics. We use original 64-step Lumina-DiMOO inference as the baseline and include direct reduction to 24 steps, denoted by +Step=24 in the tables. The remaining rows apply acceleration methods to the same pretrained model. Feature-reuse baselines include MLCache (Xin et al., 2025), LazyMAR (Yan et al., 2025), dLLM-Cache (Liu et al., 2025b), ReCAP (Liu et al., 2026), TaylorSeer (Liu et al., 2025a), and MARché (Jiang et al., 2026). We also evaluate LADR (Wang et al., 2026), FreeCorrection (Ouyang et al., 2026), Halton (Besnier et al., 2025), GtR ordering (Yan et al., 2026), and MIGM-Shortcut (Zhu et al., 2026). All accelerators use their Lumina-DiMOO editing adaptations; GtR transfers its sampling order, and MIGM-Shortcut uses its released predictor. ImgEdit is assessed with EditScore (Luo et al., 2026) and its instruction-following, consistency, and perceptual-quality components. For PIE-Bench, background PSNR, SSIM, and LPIPS are computed between the generated image and the source image within the annotated unedited region. They measure pixel fidelity, structural similarity, and perceptual distance, respectively. CLIP-Edit assesses target alignment following the regional protocol of WhereEdit (Hu et al., 2026).

Experimental environment and timing. Our workstation has two NVIDIA GeForce RTX 4090 GPUs. We report end-to-end inference latency and compute speedup by dividing the 64-step baseline's mean latency by each method's mean latency on the same benchmark. For the primary PIE-Bench timing comparison, the baseline and LightDraft run consecutively on the same GPU for each task, with balanced execution order. These measurements were collected in an earlier distributed run; the remaining timings are taken from previous benchmark runs.

4.2 COMPARISON WITH EXISTING METHODS

Editing quality and efficiency. Table 1 shows that LightDraft achieves the highest overall EditScore among the compared methods and an 8.14× speedup over the baseline. Source consistency and perceptual quality both improve over the baseline. Compared with GtR ordering, LightDraft obtains a higher overall editing score and lower latency, but a lower instruction-following score. MIGM-Shortcut is faster, but obtains a lower overall editing score.

Background preservation. On PIE-Bench (Table 2), LightDraft achieves the best background PSNR, SSIM, and LPIPS among the compared methods, with a **7.99**× speedup over the baseline. It also maintains target alignment: compared with FreeCorrection, LightDraft improves background preservation with a similar CLIP-Edit score.

Table 1: **Quantitative comparison on ImgEdit.** Subscripts denote changes from 64-step inference. Green marks improvements; bold marks the best results.

Method	Inference Efficiency		Editing Quality			
	Latency (s)↓	Speedup↑	Following↑	Consistency↑	Perceptual↑	Overall↑
Baseline (64 steps)	49.19	1.00×	5.641	6.968	5.349	4.520
+Step=24	18.06$_{-31.14}$	2.72×	5.857$_{+0.216}$	6.898$_{-0.070}$	5.140$_{-0.209}$	4.608$_{+0.089}$
+ML-Cache	6.24$_{-42.95}$	7.89×	5.021$_{-0.620}$	5.841$_{-1.128}$	3.849$_{-1.500}$	3.661$_{-0.859}$
+LazyMAR	6.48$_{-42.71}$	7.59×	5.396$_{-0.245}$	6.350$_{-0.619}$	4.455$_{-0.894}$	4.067$_{-0.453}$
+dLLM-Cache	7.35$_{-41.84}$	6.69×	5.005$_{-0.636}$	5.854$_{-1.114}$	3.849$_{-1.500}$	3.664$_{-0.856}$
+ReCAP	7.76$_{-41.43}$	6.34×	5.586$_{-0.055}$	6.482$_{-0.486}$	4.509$_{-0.840}$	4.199$_{-0.321}$
+TaylorSeer	5.55$_{-43.65}$	8.87×	1.746$_{-3.895}$	1.164$_{-5.804}$	1.104$_{-4.245}$	0.817$_{-3.703}$
+MARché	11.29$_{-37.90}$	4.36×	5.597$_{-0.043}$	6.977$_{+0.009}$	5.165$_{-0.183}$	4.397$_{-0.123}$
+LADR	9.13$_{-40.06}$	5.39×	3.858$_{-1.783}$	5.176$_{-1.792}$	3.191$_{-2.158}$	2.865$_{-1.655}$
+FreeCorrection	18.57$_{-30.63}$	2.65×	5.748$_{+0.107}$	6.891$_{-0.077}$	5.241$_{-0.107}$	4.524$_{+0.004}$
+Halton	7.01$_{-42.18}$	7.02×	5.808$_{+0.167}$	6.652$_{-0.316}$	4.776$_{-0.573}$	4.354$_{-0.166}$
+GtR ordering	7.01$_{-42.18}$	7.01×	**6.103**$_{+0.462}$	6.783$_{-0.185}$	5.028$_{-0.321}$	4.644$_{+0.124}$
+MIGM-Shortcut	**4.48**$_{-44.72}$	**10.99**×	5.434$_{-0.207}$	6.359$_{-0.610}$	4.414$_{-0.934}$	4.116$_{-0.403}$
+LightDraft (Ours)	6.04$_{-43.15}$	8.14×	5.811$_{+0.170}$	**7.317**$_{+0.349}$	**5.836**$_{+0.487}$	**4.786**$_{+0.266}$

Table 2: **Quantitative comparison on PIE-Bench.** Subscripts denote changes from 64-step inference. Green marks improvements; bold marks the best results.

Method	Inference Efficiency		Background Preservation			Edit Alignment
	Latency (s)↓	Speedup↑	PSNR (dB)↑	SSIM↑	LPIPS↓	CLIP-Edit↑
Baseline (64 steps)	48.06	1.00×	26.08	0.804	0.062	21.37
+Step=24	18.36$_{-29.70}$	2.62×	25.95$_{-0.13}$	0.804$_{0.000}$	0.061$_{0.000}$	21.45$_{+0.08}$
+ML-Cache	6.21$_{-41.85}$	7.74×	25.47$_{-0.61}$	0.801$_{-0.004}$	0.066$_{+0.004}$	21.28$_{-0.09}$
+LazyMAR	6.45$_{-41.61}$	7.45×	25.59$_{-0.49}$	0.801$_{-0.003}$	0.064$_{+0.003}$	21.35$_{-0.02}$
+dLLM-Cache	7.25$_{-40.81}$	6.63×	25.45$_{-0.63}$	0.801$_{-0.003}$	0.065$_{+0.004}$	21.16$_{-0.21}$
+ReCAP	7.73$_{-40.33}$	6.22×	25.69$_{-0.39}$	0.802$_{-0.002}$	0.063$_{+0.002}$	21.51$_{+0.14}$
+TaylorSeer	5.51$_{-42.55}$	8.72×	15.49$_{-10.59}$	0.640$_{-0.164}$	0.294$_{+0.233}$	18.81$_{-2.57}$
+MARché	11.25$_{-36.81}$	4.27×	25.74$_{-0.34}$	0.803$_{-0.001}$	0.063$_{+0.001}$	21.47$_{+0.10}$
+LADR	8.96$_{-39.10}$	5.37×	24.95$_{-1.13}$	0.798$_{-0.007}$	0.070$_{+0.009}$	20.67$_{-0.70}$
+FreeCorrection	18.49$_{-29.57}$	2.60×	26.01$_{-0.07}$	0.804$_{0.000}$	0.061$_{0.000}$	21.49$_{+0.12}$
+Halton	7.01$_{-41.05}$	6.85×	25.60$_{-0.48}$	0.799$_{-0.005}$	0.066$_{+0.004}$	21.68$_{+0.31}$
+GtR ordering	7.02$_{-41.04}$	6.85×	25.61$_{-0.47}$	0.800$_{-0.004}$	0.065$_{+0.003}$	**21.75**$_{+0.37}$
+MIGM-Shortcut	**4.45**$_{-43.61}$	**10.81**×	25.68$_{-0.40}$	0.802$_{-0.002}$	0.064$_{+0.002}$	21.27$_{-0.10}$
+LightDraft (Ours)	6.02$_{-42.04}$	7.99×	**31.28**$_{+5.20}$	**0.934**$_{+0.130}$	**0.031**$_{-0.031}$	21.49$_{+0.12}$

Qualitative comparison. Figure 4 compares our method with original inference and accelerated alternatives. Our method turns the kickboard turquoise while preserving its outline and handle opening, and inserts a folding stool that is absent from the 64-step output. For object replacement, it produces a round red ball, whereas several alternatives leave a protrusion or distort its shape. In the cabin example, our method retains a single window and the seating arrangement; several alternatives alter the window structure or extend the exterior view across surrounding panels. These examples illustrate instruction adherence and preservation of scene structure.

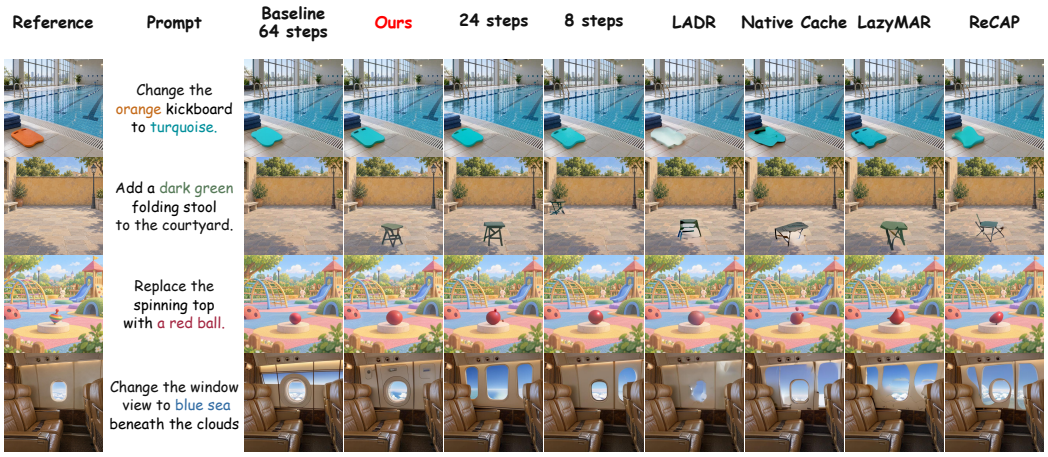


Figure 4: **Qualitative comparison of instruction-guided image editing.** Examples cover color modification, object addition, object replacement, and scene editing. Our method follows the instructions while preserving the surrounding layout. Efficiency is reported in Tables 1 and 2.

4.3 Component Analysis

Table 3 compares regional processing (R), adaptive sampling (A), and guidance-branch reuse (G) on PIE-Bench. R combines two-stage caching, minimum-support expansion, and RRC. The component controls use an earlier reference configuration. Relative to uniform step reduction, R improves background preservation while reducing latency. A reduces latency, with background and alignment scores remaining close to those of uniform step reduction. G also lowers latency, but does not yield comparable preservation gains.

Table 3: **Spatial and temporal acceleration on PIE-Bench.** R: regional caching with minimum-support expansion and RRC; A: adaptive sampling; G: guidance-branch reuse. The all-disabled control uses 24 steps. Subscripts show changes from the 64-step baseline.

R	A	G	Inference efficiency		Editing quality			
			Latency (s)↓	Speedup↑	PSNR (dB)↑	SSIM↑	LPIPS↓	CLIP-Edit↑
Baseline			48.06	1.00×	26.08	0.804	0.062	21.37
×	×	×	$18.36_{-29.70}$	2.62×	$25.95_{-0.13}$	$0.804_{0.000}$	$0.061_{0.000}$	$21.45_{+0.08}$
✓	×	×	$9.53_{-38.53}$	5.04×	$31.70_{+5.62}$	$0.934_{+0.130}$	$0.029_{-0.032}$	$21.54_{+0.17}$
×	✓	×	$10.74_{-37.32}$	4.47×	$25.90_{-0.18}$	$0.804_{0.000}$	$0.061_{0.000}$	$21.47_{+0.10}$
×	×	✓	$15.07_{-32.99}$	3.19×	$25.87_{-0.21}$	$0.804_{-0.001}$	$0.061_{0.000}$	$21.47_{+0.09}$
✓	✓	✓	$6.02_{-42.04}$	7.99×	$31.28_{+5.20}$	$0.934_{+0.130}$	$0.031_{-0.031}$	$21.49_{+0.12}$

The complete method achieves a **7.99**× speedup over the baseline while preserving background content and maintaining target alignment. The component comparisons suggest that regional processing primarily improves background preservation, while adaptive sampling reduces computation.

5 Conclusion

LightDraft uses look-ahead images to guide regional feature caching and adaptive sampling for training-free acceleration of masked diffusion image editing. It improves EditScore on ImgEdit with an 8.14× speedup and achieves a 7.99× speedup on PIE-Bench with strong background preservation. Both decisions reuse the same look-ahead images to allocate computation across spatial positions and sampling steps.

AI USE STATEMENT

In this paper we used generative AI tools solely to polish the language of this manuscript, including grammar, wording, and sentence clarity. All research ideas, experiments, results, and interpretations are entirely our own. The authors reviewed the AI-assisted edits and take full responsibility for the final content of this work.

REFERENCES

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Victor Besnier, Mickael Chen, David Hurych, Eduardo Valle, and Matthieu Cord. Halton scheduler for masked generative image transformer. In *International Conference on Learning Representations*, volume 2025, pp. 83027–83047, 2025.

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.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11305–11315. IEEE, 2022.

Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. DiffEdit: Diffusion-based semantic image editing with mask guidance. *arXiv preprint arXiv:2210.11427*, 2022.

Yingying Deng, Xiangyu He, Changwang Mei, Peisong Wang, and Fan Tang. FireFlow: Fast inversion of rectified flow for image semantic editing. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 13110–13128. PMLR, 2025. URL https://proceedings.mlr.press/v267/deng25c.html.

Gilad Deutch, Rinon Gal, Daniel Garibi, Or Patashnik, and Daniel Cohen-Or. TurboEdit: Text-based image editing using few-step diffusion models. *arXiv preprint arXiv:2408.00735*, 2024. URL https://arxiv.org/abs/2408.00735.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Ming Hu, Mingyu Dou, Jianfu Yin, Miaomiao Zhang, Cong Hu, Yao Wang, Bingliang Hu, and Quan Wang. Whereedit: Mask-aware local latent editing for one-step image editing. *arXiv preprint arXiv:2607.20883*, 2026.

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.

Chaoyi Jiang, Sungwoo Kim, Lei Gao, Hossein Entezari Zarch, Won Woo Ro, and Murali Annavaram. March
'e: Fast masked autoregressive image generation with cache-aware attention. In *European Conference on Computer Vision*, pp. 624–640. Springer, 2026.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Direct inversion: Boosting diffusion-based editing with 3 lines of code. *arXiv preprint arXiv:2310.01506*, 2023.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Shufan Li, Jiuxiang Gu, Kangning Liu, Zhe Lin, Zijun Wei, Aditya Grover, and Jason Kuen. LaViDa-O: Elastic large masked diffusion models for unified multimodal understanding and generation. In *International Conference on Learning Representations*, volume 2026, pp. 50571–50603, 2026a.

Shufan Li, Jiuxiang Gu, Kangning Liu, Zhe Lin, Zijun Wei, Aditya Grover, and Jason Kuen. SparseLaViDa: Sparse multimodal discrete diffusion language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36103–36114, 2026b.

Jiacheng Liu, Chang Zou, Yuanhuiyi Lyu, Junjie Chen, and Linfeng Zhang. From reusing to forecasting: Accelerating diffusion models with taylorseers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15853–15863. IEEE, 2025a.

Xuejie Liu, Anji Liu, Guy Van den Broeck, and Yitao Liang. Plug-and-play context feature reuse for efficient masked generation. *Advances in Neural Information Processing Systems*, 38:32352–32379, 2026.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dLLM-Cache: Accelerating diffusion large language models with adaptive caching. *arXiv preprint arXiv:2506.06295*, 2025b. URL https://arxiv.org/abs/2506.06295.

Xin Luo, Jiahao Wang, Chenyuan Wu, Shitao Xiao, Xiyan Jiang, Defu Lian, Jiajun Zhang, Dong Liu, and Zheng Liu. Editscore: Unlocking online rl for image editing via high-fidelity reward modeling. In *International Conference on Learning Representations*, volume 2026, pp. 33027–33056, 2026.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. DeepCache: Accelerating diffusion models for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15762–15772, 2024.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6038–6047. IEEE, 2023.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025. URL https://arxiv.org/abs/2502.09992.

Yidong Ouyang, Panwen Hu, Zhengyan Wan, Zhe Wang, Liyan Xie, Dmitriy Bespalov, Ying Nian Wu, Guang Cheng, Hongyuan Zha, and Qiang Sun. Training-free self-correction for multimodal masked diffusion models. *arXiv preprint arXiv:2602.02927*, 2026.

Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of GANs for semantic face editing. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9240–9249. IEEE, 2020.

Chenglin Wang, Yucheng Zhou, Shuang Chen, Tao Wang, and Kai Zhang. Ladr: Locality-aware dynamic rescue for efficient text-to-image generation with diffusion large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27170–27183, 2026.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004. doi: 10.1109/TIP.2003.819861. URL https://doi.org/10.1109/TIP.2003.819861.

Qingyan Wei, Yaojie Zhang, Zhiyuan Liu, Puyu Zeng, Yuxuan Wang, Biqing Qi, Dongrui Liu, and Linfeng Zhang. Accelerating diffusion large language models with SlowFast sampling: The three golden principles. *arXiv preprint arXiv:2506.10848*, 2025. URL https://arxiv.org/abs/2506.10848.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. *arXiv preprint arXiv:2505.22618*, 2025. URL https://arxiv.org/abs/2505.22618.

Yi Xin, Qi Qin, Siqi Luo, Kaiwen Zhu, Juncheng Yan, Yan Tai, Jiayi Lei, Yuewen Cao, Keqi Wang, Yibin Wang, et al. Lumina-DiMOO: An omni diffusion large language model for multi-modal generation and understanding. *arXiv preprint arXiv:2510.06308*, 2025.

Feihong Yan, Qingyan Wei, Jiayi Tang, Jiajun Li, Yulin Wang, Xuming Hu, Huiqi Li, and Linfeng Zhang. Lazymar: Accelerating masked autoregressive models via feature caching. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15552–15561. IEEE, 2025.

Feihong Yan, Yao Zhu, Peiru Wang, Qingyan Wei, Huiqi Li, Linfeng Zhang, et al. Generation then reconstruction: Accelerating masked autoregressive models via two-stage sampling. In *International Conference on Learning Representations*, volume 2026, pp. 134229–134243, 2026.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Mmada: Multimodal large diffusion language models. *Advances in Neural Information Processing Systems*, 38:138867–138907, 2026.

Yang Ye, Xianyi He, Zongjian Li, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, Li Yuan, et al. Imgedit: A unified image editing dataset and benchmark. *Advances in Neural Information Processing Systems*, 38, 2026.

Xinjie Zhang, Peng Zhang, Shicheng Zheng, Jinghao Guo, Zhaoyang Jia, Yifei Shen, Xun Guo, Yuxuan Luo, Jiahao Li, Wenxuan Xie, et al. Mage-Flow: An efficient native-resolution foundation model for image generation and editing. *arXiv preprint arXiv:2607.19064*, 2026.

Kaiwen Zhu, Quansheng Zeng, Yuandong Pu, Shuo Cao, Xiaohui Li, Yi Xin, Qi Qin, Jiayang Li, Yu Qiao, Jinjin Gu, et al. Accelerating masked image generation by learning latent controlled dynamics. *arXiv preprint arXiv:2602.23996*, 2026.