EEdit⚡: Rethinking the Spatial and Temporal Redundancy for Efficient Image Editing

Anonymous ICCV submission

Paper ID 4183

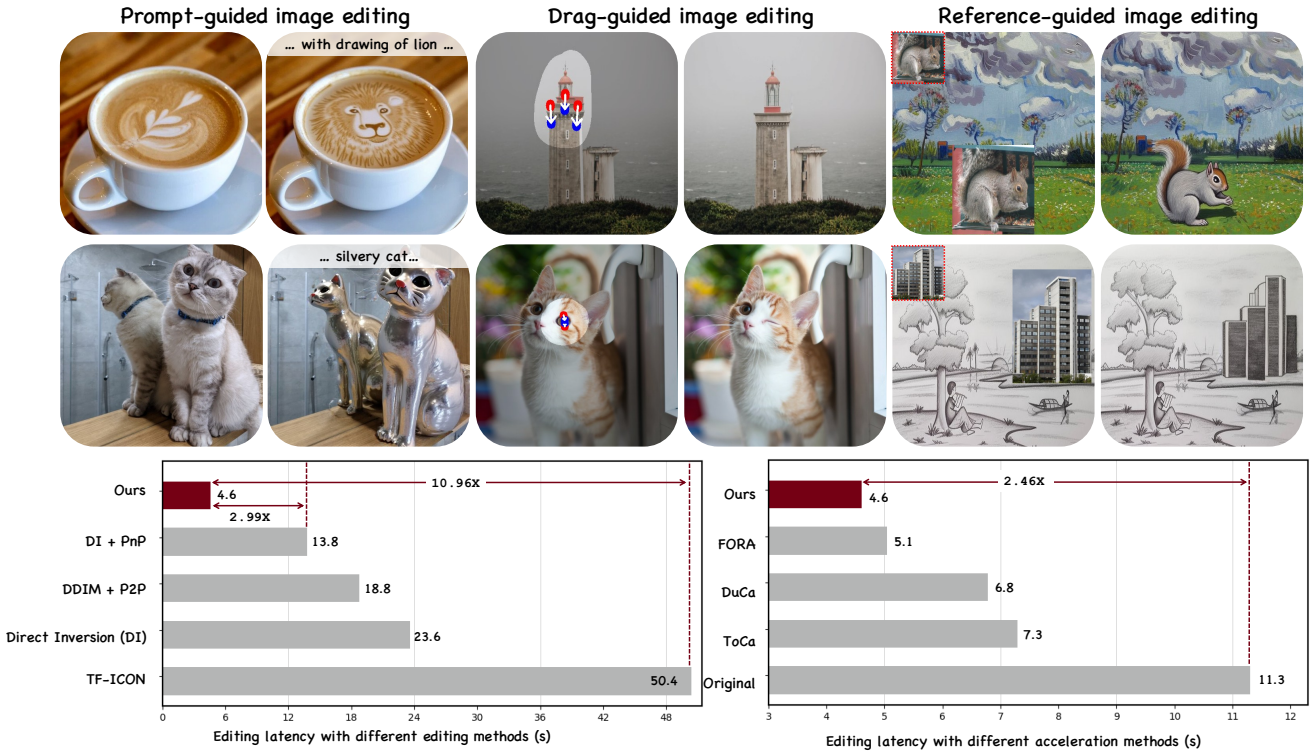

Figure 1. **Gallery of various editing tasks and efficiency comparisons.** We propose the EEdit, a novel inversion-based framework for efficient image editing. Compare with previous methods, we achieve the faster and more efficient image editing.

Abstract

Inversion-based image editing is rapidly gaining momentum while suffering from significant computation overhead, hindering its application in real-time interactive scenarios. In this paper, we rethink that the redundancy in inversion-based image editing exists in both the spatial and temporal dimensions, such as the unnecessary computation in unedited regions and the redundancy in the inversion progress. To tackle these challenges, we propose a practical framework, named ***EEdit****, to achieve efficient image editing. Specifically, we introduce three techniques to solve them one by one.* ***For spatial redundancy****, spatial locality caching is introduced to compute the edited region and its neighboring regions while skipping the unedited regions, and token indexing preprocessing is designed to further accelerate the caching.* ***For temporal redundancy****, inversion step skipping is proposed to reuse the latent for efficient editing. Our experiments demonstrate an average of* ***2.46****× acceleration without performance drop in a wide range of editing tasks including prompt-guided image editing, dragging and image composition. Our codes are available in the supplementary material and will be released on Github.*

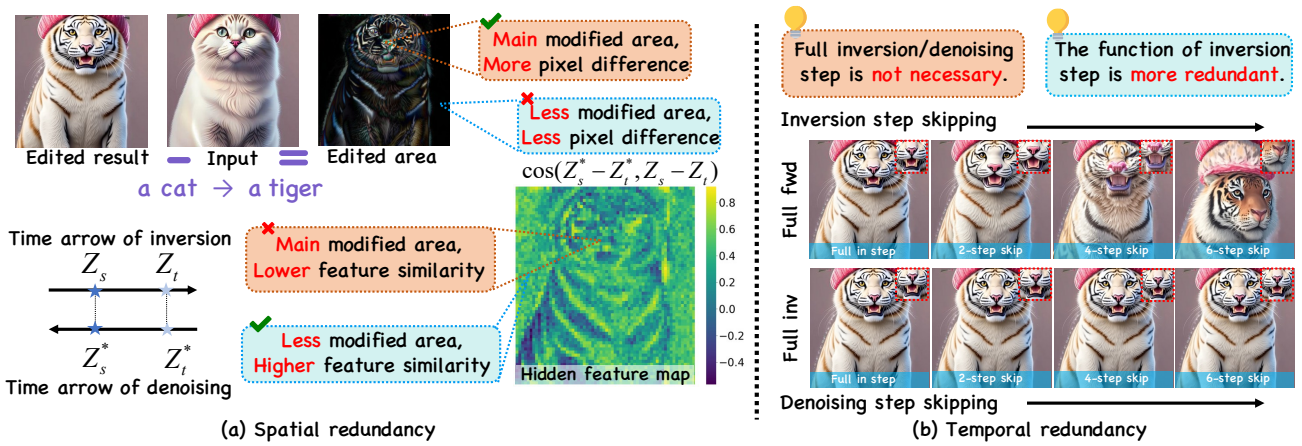

Figure 2. **The illustration of motivation**. We observe that there are significant computational redundancy during the editing process, which can be classified into spatial redundancy (a) and temporal redundancy (b). For spatial redundancy (a), due to a high background similarity between the source image and the edited results, calculating the entire image in every inversion/denoising step is unnecessary. For temporal redundancy (b), we discover that skipping some steps in inversion/denoising does not degrade the editing performance.

1. Introduction

With groundbreaking advances in diffusion models [6, 8, 30, 44], they have emerged as the state-of-the-art approach for both image generation and editing tasks [4, 11, 13, 45]. Current diffusion-based editing frameworks typically adopt a two-stage pipeline: (1) an *inversion process* [27, 29, 33, 41] that maps the input image to its corresponding noisy latent, followed by (2) a *denoising process* [16, 24, 33, 38] that progressively denoises and modifies the latent code to produce the edited image. While achieving impressive editing quality, this two-stage paradigm suffers from significant computational overhead, particularly hindering its usage from real-time interactive applications on resource-constrained edge devices. To address this problem, this paper begins by identifying two types of computational redundancy in editing and then solves them one by one.

(I) Spatial Redundancy: Although only a small region of the image is expected to be edited, the current editing pipeline has to compute all the pixels of the given image. Retaining computation in these unedited regions incurs additional computational overhead, while yielding minimal benefits for our editing objectives. In Figure. 2, we visualize and compare the differences between pixels before and after image editing, as well as the heat map of cosine similarity of the latent space state differences at fixed time intervals. Both patterns consistently exhibit spatial redundancy differences in the editing task: the edited regions show greater pixel differences and lower similarity in the latent space, whereas the unedited regions exhibit negligible pixel differences and higher similarity in the latent space. Specifically, the unedited region exhibits significantly greater redundancy compared to the edited region.

(II) Temporal Redundancy: Compared with generating a new image, image editing poses additional computation cost for inversion, which is employed to map the to-be-edited image to the latent space of noise to some extent. In the traditional editing pipeline, the inversion process takes the same computation cost as denoising and thus doubles the computational overhead. In Figure. 2, we separately control the full inversion and denoising processes by introducing interval-skipping time steps during the inversion/denoising process. It allows us to compare how reducing the computational load of one process under the same overall computational cost affects the editing performance. Notably, reducing the denoising steps first led to the loss of fine details, such as texture information around the mouth and nose, and soon resulted in the degradation of overall structural integrity. Surprisingly, skipping time steps in the inversion process had little to no perceptible effect, indicating a highly unbalanced distribution of temporal redundancy between inversion and denoising. Specifically, we discover that inversion exhibits significantly greater redundancy compared to denoising.

Based on this observation, we propose **S**patial **Lo**cality **C**aching (**SLoC**), which aims to skip the computation of unedited regions by reusing their features computed in the previous timesteps. Concretely, during both the denoising and inversion process, SLoC first computes the tokens corresponding to all the regions and stores them in a cache. Then, in the following step, SLoC still performs full computation on the edited region and its neighboring regions, while skipping the computation of other regions by reusing their previously cached features. Such a mixed-computation manner in SLoC enables diffusion models to pay more effort in important regions guided by the editing

prior, maintaining good quality in the edited region while trading the computation in unedited region for efficiency. Furthermore, to further reduce the computation from SLoC, we analyze that caching initialization and update strategy for the score map can be finished as an offline operation. Based on this observation, we propose the token index preprocessing, achieving over a **15%** improvement in inference speed. Importantly, this optimization is mathematically equivalent, ensuring that SLoC itself undergoes an additional lossless acceleration.

Additionally, we observe that there is temporal redundancy in both inversion/denoising processing. Motivated by DDIM [38], we propose the inversion step skipping strategy. The primary insight is that *skipping certain inversion steps does not produce noticeable artifacts and can markedly improve processing speed.* Surprisingly, our experiments reveal that the number of timesteps for inversion can be safely reduced to 33.3% of that for diffusion, with almost no noticeable performance degradation.

We validated the effectiveness of proposed modules in extensive experiments on various editing tasks, including prompt-guided [6, 13, 42], reference-guided [24, 43, 46], and drag-guided editing [22, 28, 36, 48]. While achieving state-of-the-art performance in background consistency, EEdit also delivers up to **10.96**× latency acceleration compared to other editing methods. Even when compared to existing cache-based acceleration techniques, it demonstrates a superior acceleration ratio. Furthermore, EEdit achieves a **2.46**× speedup with minimal performance differences, maintaining near-lossless quality across various evaluation metrics when compared to the original editing pipeline without cache acceleration.

In summary, our contributions include:

- We emphasize the spatial and temporal redundancy in inversion-based editing tasks and propose EEdit, a novel editing framework to modify images efficiently.
- **Spatial Locality Caching (SLoC):** To reduce the spatial redundancy in editing, spatial locality caching is proposed to skip most of the computation in unedited regions. Then we design the **T**oken **I**ndex **P**reprocessing **(TIP)** to further optimize the speed of caching.
- **Inversion Step Skipping (ISS):** To reduce the temporal redundancy, we propose to assign more computation to denoising and fewer computation to inversion, which firstly demonstrates and leverages the unequal importance of the two processes to accelerate editing.
- **Extensive Adaption and Experiments:** We evaluate our methods across various editing tasks, including prompt-guided editing, drag-guided editing, and reference-guided editing, which demonstrates 10.96× acceleration than the current state-of-the-art editing approach.

2. Related Work

2.1. Acceleration of Diffusion Models

Low-latency, high-quality generation methods are an important research direction. Currently, there are two main types of diffusion model acceleration methods: the first one is to reduce the number of sampling steps [19–21, 38], and the second one is to accelerate internal computation of the diffusion model. Solutions to reduce computational complexity include model distillation and compression [44], token pruning [47], token merging [2, 3], and layer-wise caching techniques [18, 25, 26, 35]. However, layer-wise caching techniques have a large cache granularity, which ignore the asymmetry of importance at the token level. Toca [50] and Duca [51] adopt a token-wise cache, focusing on and assigning importance to tokens with score maps. During recomputation, a certain proportion of tokens are selected for refreshing, achieving a lossless acceleration.

Unfortunately, existing caching schemes are not specifically designed to accommodate the characteristics of editing tasks and suffer from the following issues:

First, computing intrinsic feature correlations between tokens introduces additional internal overhead in caching operations. Second, some caching strategies and editing methods require accessing attention maps [50] or storing KV matrices [24, 49], leading to incompatibility with existing Transformer acceleration techniques, such as FlashAttention [7], thereby increasing latency. Moreover, editing tasks inherently exhibit prior information regarding the spatial distribution of regions requiring greater attention, yet current caching techniques fail to leverage this information.

Our approach significantly improves acceleration over traditional caching methods by effectively reducing both temporal and spatial redundancy. Additionally, we introduce token index preprocessing to further compress the extra overhead incurred by caching. As a result, our method simultaneously optimizes both editing performance and acceleration efficiency among various editing tasks.

2.2. Image Editing

Image editing, as an important application in the generative field, has received widespread research and exploration in the academic community. This includes controllable text-to-image generation, image inpainting, and image-to-image generation, among other approaches [6, 9, 10, 14, 31, 34]. Common editing approaches follow a noise addition and denoising framework, where the original image is perturbed by a certain level of noise in the latent space to leverage the model's editing capabilities. The final edited image is then obtained through a denoising process that restores clarity. Training-free inversion editing techniques, when applied to editing tasks such as prompt-guided editing, image composition, and image dragging, involve operations on the at-

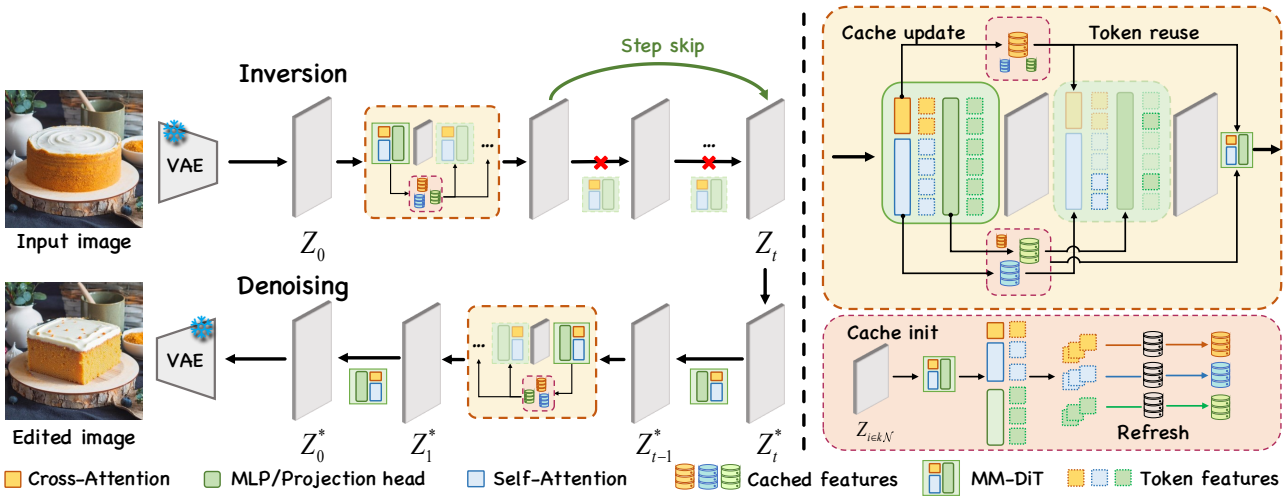

Figure 3. **The overview of our approach**. The proposed framework for image editing based on MM-DiT diffusion models employs an efficient denoising and training-free approach. The pipeline takes the original image and an editing prompt as inputs. Specifically, the cache is refreshed entirely in fixed time-step interval, while partial computation for updating cache is maintained for the intermediate timesteps.

tention map, including modification, enhancement, and replacement. These methods have been applied in P2P [11] and more subsequent works [24, 49]. Since inversion-based approaches require the inversion technique to reconstruct the denoised image and significantly impact editing quality, researchers have explored both training-based and training-free inversion techniques. InfEdit adopts a virtual inversion strategy without explicit inversion during sampling, enabling accurate and consistent editing, and falls under the category of inversion-free image editing methods. Unfortunately, these methods based on or utilizing attention maps require storing the attention map and manipulating the corresponding KV matrices [11, 49]. This results in incompatibility with common attention acceleration techniques, increasing inference latency. In contrast, our method enables reduces the redundancy in inversion and denoising, enhancing the efficiency and speed of editing.

3. Preliminaries

3.1. Rectified Flow

The Rectified Flow [17] method models the transformation from a Gaussian noise distribution π_0 to the real data distribution π_1 as a continuous change along a straight path by learning a forward simulation system.

In the forward process, the state of the system can be viewed as a linear interpolation between the initial state and the Gaussian noise, which is simplified and easier to understand compared to traditional methods [12].

$$\mathbf{X}_t = (1-t)\mathbf{X}_1 + t\mathbf{X}_0, \quad X_1 \sim \pi_1, X_0 \sim \pi_0 \tag{1}$$

By differentiating the above expression with respect to t, the ODE form of the Rectified Flow can be obtained: $\frac{d\mathbf{X}_t}{dt} = \mathbf{X}_1 - \mathbf{X}_0$. Furthermore, let us define $v_\theta = \frac{dX_t}{dt}$, the training objective can then be transformed into minimizing the integral of this expectation over the time steps:

$$\min_\theta \int_0^1 \mathbb{E}\left[\|\mathbf{X}_1 - \mathbf{X}_0 - v_\theta(\mathbf{X}_t, t)\|^2\right] dt \tag{2}$$

Here, θ represents the parameters of the neural network to be optimized. Due to the complexity introduced by the integral symbol, the optimization of these parameters is typically performed using the equivalent form of Conditional Flow Matching (CFM):

$$\mathcal{L}_{CFM} = \mathbb{E}_{t,p_t(z|\epsilon),p(\epsilon)} \|v_\Theta(z,t) - u_t(z|\epsilon)\|_2^2, \tag{3}$$

where the conditional vector fields $u_t(z|\epsilon)$ provides an equivalent yet tractable objective. p_t is the probability path between p_0 and p_1, and $p_1 \sim \pi_0$.

4. Approaches

Our approaches aim to reduce the spatial and temporal redundancy to improve the efficiency of image editing. The core idea is to leverage the mask of edited area to guide cache refreshing and reuse. The pipeline of the EEdit is shown in Figure. 3. In this section, we will describe the design of spatial locality caching (see Section. 4.1) and token index preprocessing (see Section. 4.2) for spatial redundancy. For temporal redundancy, we introduce the inversion step skipping in the Section. 4.3.

4.1. Spatial Locality Caching

Score Bonus for Editing Region. In mask-guided image editing tasks, the tokens corresponding to image patches

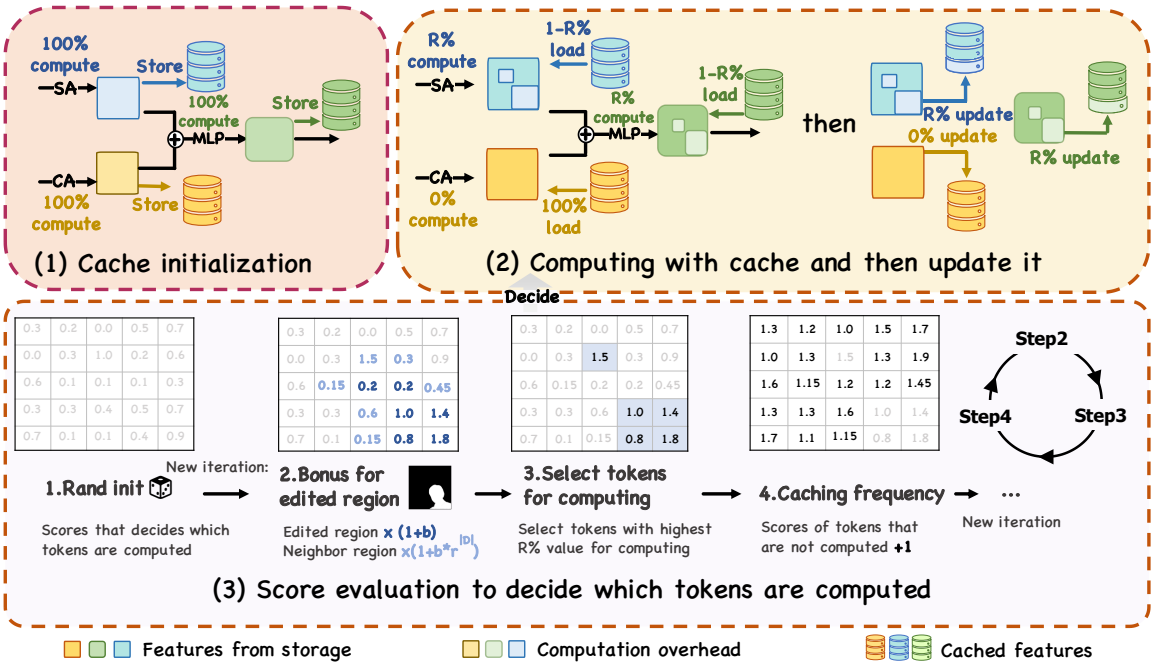

Figure 4. **The pipeline of spatial locality caching.** (1) The initialization and refresh process of cache storage using the computed results from SA (Self-Attention), CA (Cross-Attention), and MLP. (2) The token-wise partial computation logic and the cache update mechanism. (3) The initialization and update logic for scoring, which is responsible for selecting indices for partial computation.

outside the edited region are actually derived from Z_T^{inv}, which is obtained from inversion process. This part is directly replaced in the latent space.

As a result, the importance of DiT's computation on these tokens is significantly reduced. Instead, we aim to focus more on the computation of tokens within the edited region. Simultaneously, to account for the modeling of correlations between important tokens and their neighboring tokens, the neighboring tokens are rewarded with gradually decreasing importance. We designed a score bonus map $\mathbf{S_E}$ to control the reward intensity, defined as:

$$\mathbf{S_E}(\mathbf{x}) = \begin{cases} 1 + b \cdot r^k, & \mathbf{x} \in \mathcal{N}_k(M_s),\ k \in 0, 1, \ldots, K \\ 1, & \mathbf{x} \notin \bigcup_{k=0}^{K} \mathcal{N}_k(M_s) \end{cases}$$

Here $\mathbf{x}$ represents an arbitrary token in DiT. M_s is the latent mask for point set of edit region. Hyperparameters such as $b > 1$ represent the bonus factor, while $0 < r < 1$ denotes the decay ratio. K is the maximum number of layers for the neighborhood. The neighborhood of M_s with an L1 norm equal to k is denoted as $\mathcal{N}_k(M_s)$ and is defined as:

$$\mathcal{N}_k(M_s) = \{\mathbf{x} : \exists \mathbf{e} \in M_s, \|\mathbf{x} - \mathbf{e}\|_1 = k\}.$$

Update Strategy with Cache Frequency Control. When the proportion of tokens corresponding to the edited region is lower than the full computation of the network, meaning it is smaller than the cache refresh ratio, not all tokens in the edited region undergo the same level of full computation. Instead, we prioritize recomputing and refreshing in the cache those tokens that have been updated less frequently and reused more times.

SLoC track the frequency of times each token is reused from the cache and increment its score accordingly on a token-wise map $\mathcal{M}_{freq}$. The more frequently a token is reused, the more likely its features will be recomputed and refreshed. Once certain tokens undergo recomputation and refresh, their corresponding usage frequency counters are immediately reset to zero. From another perspective, the design of cache frequency control serves two key purposes. First, it encourages the recomputation of frequently reused tokens to reduce accumulated errors. Second, it suppresses the redundant recomputation of tokens that are repeatedly updated, thereby reducing computational overhead.

4.2. Token Index Preprocessing

Although SLoC reduces spatial redundancy by using caching, as a double-edged sword, such an acceleration comes with certain limitations: in a cache cycle shown in Figure. 4, the cache overhead is sequentially arranged as follows: (1) Randomly initialize. (2) Bonus for edited region. (3) Perform sorting and selection. (4) Update. (5) Refresh. These steps constitute the caching cycle and operate as illustrated in Figure. 4.

However, we observe that internal score updates and token selection take additional computational costs in editing

Algorithm 1 SLoC Editing with **ISS&TIP**

Require: Input image $\mathbf{I_s}$, Mask for editing region $\mathbf{M_s}$, Prompt for editing $\mathbf{P_m}$, Randomly initialized map $\mathcal{R}$, Bonus for edited region $\mathbf{S_E}$ and Cache dict $\mathcal{C}_{l,m}[:]$.
Ensure: The edited result $\mathbf{I}^*$

// **Token Index Preprocessing**
$\mathcal{M}_{freq} \leftarrow \boldsymbol{zero}[\mathcal{F}_\mathbf{i}, \mathbf{l}]$
for $t = T, T-1, \ldots 1$ **do**
  for $\mathcal{F}_i \leftarrow \mathbf{SA}_l, \mathbf{CA}_l, \mathbf{MLP}_l, l \in [1, 2 \ldots \mathcal{L}]$ **do**
    $\mathcal{S}_l \leftarrow (\mathcal{R} \odot \mathbf{S_E}) \oplus \mathcal{M}_{freq}$
    $\mathcal{I}_{i,l,t} \leftarrow \mathbf{Sel}_{topR\%}(\mathcal{S}_l)$
    $\mathbf{Update}(\mathcal{M}_{freq})$
  end for
end for
$\mathbf{Z}_0 \leftarrow \mathbf{cat}(\text{VQ-Encoder}(\mathbf{I_s}), \text{Txt-Encoder}(\mathbf{P_m}))$
// **Inversion Reduction**
for $t = 1, 2 \ldots, T$ **do**
  $\mathbf{Z}_t \leftarrow \begin{cases} \mathbf{Z}_{t-1} & \textbf{if } t \bmod m \neq 1 \textbf{ and } m \neq T, m \in \mathcal{N} \\ \text{RF-inversion}(\mathbf{Z}_{t-1}, t-1, \phi) & \textbf{otherwise} \end{cases}$
end for
$\mathbf{Z}_T^* \leftarrow \mathbf{Z}_T$
// **Image Editing Steps with SLoC**
for $t = T, T-1, \ldots 1$ **do**
  for $\mathcal{F}_i \leftarrow \mathbf{SA}_l, \mathbf{CA}_l, \mathbf{MLP}_l, l \in [1, \ldots \mathcal{L}]$ **do**
    $Z_{l+1}^* \leftarrow \mathbf{scatter}(\mathcal{F}_i(Z_l^*, \mathcal{I}_{i,l,t}), \mathcal{C}_{t+1}[l, \mathcal{F}_i])$
    $\mathbf{Update}(\mathcal{C}_{t+1}[l, \mathcal{F}_i])$
  end for
  $\mathbf{Z}_{t-1}^* \leftarrow \mathbf{Z}_{t-1}^* \odot \mathbf{M}_s + \mathbf{Z}_t \odot (1 - \mathbf{M}_s)$
end for
$\mathbf{I}^* \leftarrow \text{VQ-Decoder}(\mathbf{Z}_0^*)$
return $\mathbf{I}^*$

processing and can be further optimized. Our initialization and update strategy for the score map can be transformed from an online operation into an offline algorithm while maintaining full mathematical equivalence (See the supplementary material for the formal proof). The key insight here is that under the score update rule

$$\mathcal{S} \leftarrow (\mathcal{R} \odot \mathbf{S_E}) \oplus \mathcal{M}_{freq},$$

we can prove the top-$R\%$ selected indices $\mathcal{I}^{(t)}_{\text{topR\%}}$ in the same time step in the offline and online process remain equivalent:

$$\mathcal{I}^{(t)}_{\text{topR\%}}(\text{offline}) = \mathcal{I}^{(t)}_{\text{topR\%}}(\text{online}) \quad \forall t \in [1 \ldots T].$$

Here $\mathcal{S}$ denotes the score map, $\mathcal{R}$ represents the randomly initialized score values, $\mathbf{S_E}$ corresponds to the region score bonus, and $\mathcal{M}_{freq}$ denotes the matrix for cache frequency control, consistent with Algorithm 1.

Therefore, we can decouple the cache score update, sorting, and index selection logic from the model's computation process. By precomputing and storing the required token indices during preprocessing, the overhead from cache operations in the inference phase is reduced to a single read/write cost. Consequently, the former 4 steps of the cache cycle can be omitted during inference. SLoC directly updates only the necessary tokens during inference, without executing any redundant score computation or update strategies.

4.3. Inversion Step Skipping

After addressing the spatial redundancy, we focus on the temporal redundancy in both inversion/denoising processing. As demonstrated in Fig. 2(b), previous methods [11, 33] calculate the noise at each timestep. However, the emergence of DDIM [38] motivates us to consider *whether we can skip certain steps in flow matching-based inversion*. During the editing process, we discover that skipping some inversion steps does not degrade the quality of the editing results(see Fig. 2(b)), which demonstrates the redundancy of the inversion process.

To eliminate this redundancy, we propose the inversion step skipping strategy in the editing processing. The key insight of inversion step skipping is that *skipping some inversion steps does not result in noticeable artifacts, but it can significantly improve speed*. Formally, during the inversion, the skipping step is set m. We add the noise from the $\mathbf{Z}_0$. In every m steps, we perform an rf-inversion [33], while for the other timesteps, we directly reuse the noise from the previous timestep. This process can be formally expressed as follows:

$$\mathbf{Z}_t \leftarrow \begin{cases} \mathbf{Z}_{t-1} & \textbf{if } t \bmod m \neq 1 \textbf{ and } m \neq T, m \in \mathcal{N} \\ \text{RF-inversion}(\mathbf{Z}_{t-1}, t-1, \phi) & \textbf{otherwise} \end{cases}$$

Additionally, we provide the pseudocode implementation in Algorithm. 1. Note that, to ensure noise quality, the final step of the inversion process is always fully computed, whereas the intermediate steps employ a mixed strategy of full computation and cache-accelerated computation. In our experiment, skipping step m is set 3 to balance the quality and speed. We also provide the ablation study about the inversion skipping step in Section. 5.3.

5. Experiments

5.1. Implementation Details

In our experiment, the base model adopted the currently popular flow matching model, FLUX-Dev [15], which consists of 12B parameters. Models for other qualitative results are implemented using SD series [1, 5, 32, 37] with original codebase. Our inference and editing pipeline is built upon the codebase from Hugging Face. The inversion and denosing step is set to 28. RF-inversion relative hyper-parameters follow original implementations [33]. We employ the text-guidance ratio of 7.0. We selected the PIE-Bench [39] Benchmark as the dataset for prompt-guided editing, the

Figure 5. **Qualitative comparisons between baselines and our approach**. We compare our approach with various editing tasks, including prompt-guided image editing, drag-guided image editing, and reference-guided image editing

TF-ICON Test Benchmark [23] as the dataset for reference-guided editing, and DragBench-SR and DragBench-DR as the datasets [40] for drag-guided editing. All experiments were conducted on an NVIDIA H20. More details and evaluation metrics can be found in supplementary material.

5.2. Compare with Baseline

Qualitative Comparison. We perform extensive qualitative comparisons between our method and various editing approaches. Specifically, we evaluate three kinds of popular editing tasks, including prompt-guided, drag-guided, and reference-guided image editing. (1) **Prompt-guided image editing**: we compare with Direct Inversion [13], InfEdit [45], and P2P [11]. (2) **Drag-guided image editing**, we select three methods to do comparisons, including DragDiffusion [36], Region Drag[22], and FastDrag [48]. Pipeline are implemented from public codebase from github. (3) **Reference-guided editing**, we compare our approach with TF-ICON [24], PrimeComposer [43], and DCCF [46]. SD-v2-1 checkpoint is used for Image Composition. As shown in Figure. 5, for prompt-guided task, Direct inversion and InfEdit fail to edit the image successfuly. DDIM+P2P has a challenge to maintain the background consistency. By contract, our approach exhibits superior consistency, enhanced detail preservation, and improved aesthetic quality in terms of style.

Quantitative Comparison. We compared various editing methods in terms of background consistency in non-edited regions and adherence to prompts in the edited regions. As shown in Table. 1, since our approach allocates fewer computational resources to non-edited regions and employs masking in the latent space, we achieve significantly better performance in background consistency compared to other state-of-the-art editing methods. Furthermore, in terms of prompt adherence, the Inversion step skipping strategy achieves a CLIP score comparable to that of RF-Inversion with full computation. It is also competitive with others.

Table 1. **Comparison of quality** across different editing methods.

Editing methods	Inversion type	PSNR ↑	LPIPS $\downarrow_{\times 10^{-2}}$	SSIM ↑	CLIP-T ↑
P2P [11]	DDIM	17.87	20.88	0.72	25.13
MasaCtrl [4]	DDIM	22.19	10.54	0.80	24.02
	DI	22.69	8.73	0.82	24.39
PnP [13]	DDIM	25.23	11.27	0.80	25.42
	DI	22.46	10.55	0.80	25.48
InfEdit [45]	Inversion-free	28.11	5.61	0.85	**25.86**
DiT4Edit [9]	DPM-Solver++	22.85	-	-	25.39
SLoC	RF-inversion	31.97	1.96	0.94	25.37
SLoC	ISS	**31.97**	**1.95**	**0.94**	25.38

5.3. Ablation Study

Extensive quantitative and qualitative ablation experiments were conducted to analyze the impact of our methods on editing quality. We measure the similarity to the input im-

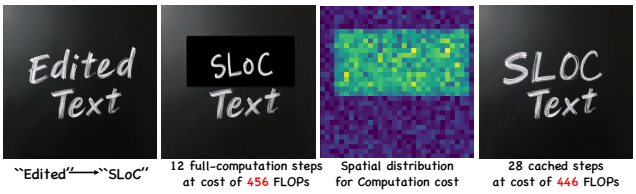

Figure 6. **A qualitative example of ablation over SLoC**, demonstrating that regional computation with SLoC leads to higher quality and lower computational overhead.

Figure 7. **The qualitative ablation study about inversion step skipping.** We visualize results in various inversion step skipping.

age in non-edited regions as *background preservation (BG preservation)*. Additionally, we compare the editing results incorporating various modules with the editing results obtained from full computation without caching as *foreground fidelity (FG fidelity)*.

Ablation study of spatial locality caching. Figure. 6 illustrates the impact of SLoC on editing performance. In text editing tasks, SLoC leverages the prior of spatial locality to achieve superior editing results compared to the original full-computation editing method, despite using more steps but incurring lower computational overhead. This validates the feasibility of our regionally focused computation strategy. Furthermore, quantitative comparisons with other cache-based acceleration methods demonstrate the superiority of our approach (shown in supplementary material).

Table 2. **Ablation study on ISS**. Ablation study is conducted on different skipping settings for background preservation, foreground fidelity and inference time.

Inversion	Denoising	BG preservation	FG fidelity			Inference ↓
		LPIPS ↓ $_{\times 10^{-2}}$	LPIPS ↓ $_{\times 10^{-3}}$	PSNR ↑	FID↓	Time (s)
Full step	**Full step**	1.98	-	-	-	13.27
2-step skip	**Full step**	1.98	5.46	43.77	3.35	10.16
3-step skip		1.98	5.29	43.99	3.23	9.31
4-step skip		1.98	5.29	43.80	3.31	8.76

Ablation study of inversion step skipping. It can be observed in Table. 2 and Figure. 7 that increasing the skip interval m to reduce inversion computation significantly decreases inference time while maintaining background preservation without quality degradation. Additionally, the foreground fidelity metrics exhibit negligible differences compared to full-step computation. This validates the effectiveness and fidelity of our ISS strategy.

Ablation study of TIP over ISS and various tasks. As

Table 3. **Ablation study on different configurations of TIP and ISS** in prompt-guided, drag-guided, and ref-guided editing.

TASK	Method		FG fidelity			Inference (s) ↓	CLIP-E ↑
	TIP	ISS	FID↓	PSNR↑	LPIPS↓ $_{\times 10^{-2}}$		
Prompt	×	×	39.50	31.75	5.75	5.96	21.34
	×	✓	39.33	31.76	5.74	5.06	21.34
	✓	×	39.42	31.75	5.75	5.14	21.34
	✓	✓	**39.21**	**31.76**	**5.74**	**4.60**	**21.34**
Dragging	×	×	**20.61**	33.47	2.28	7.12	22.20
	✓	✓	22.07	**33.68**	**2.19**	**5.66**	**22.21**
Composition	×	×	**12.33**	39.78	0.54	7.25	23.35
	✓	✓	12.35	**39.80**	**0.54**	**5.66**	**23.35**

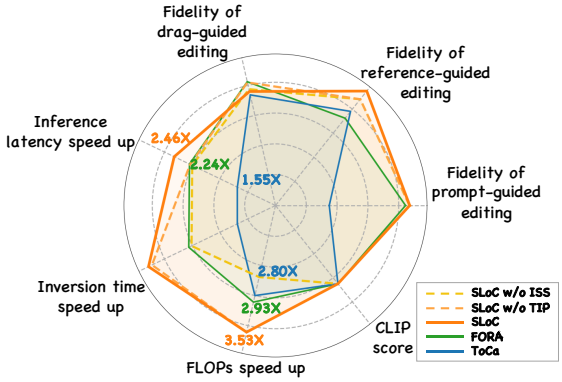

Figure 8. **A Comparison over different configurations and other cache methods.** Fidelity of various tasks, speed up ratio and clip score are shown in this radar chart.

shown in Table 3 and Figure 8, TIP remains compatible with ISS while maintaining comparable FG fidelity and CLIP scores across various editing tasks. Furthermore, it achieves an additional reduction in inference latency by an average of over 20% compared to the SLoC baseline. Under the same configuration, the impact of incorporating TIP on editing quality remains within the range of random fluctuations. Furthermore, our quantitative experiments validate that this approach achieves lossless acceleration while maintaining the same generation quality which can be seen in supplementary material.

6. Conclusions

Inversion-based image editing usually suffers from expensive computation costs caused by the spatial and temporal redundancy within diffusion models. To solve this problem, we design an efficient editing framework using the spatial locality caching with token index preprocessing and inversion step skipping. To the best of our knowledge, we are the first to adapt cache-based acceleration for diffusion inference across various editing tasks. Our work provides valuable insights and exploration for future approaches toward efficient, and potentially real-time, image and even video editing.

References

[1] Stability AI. Stable diffusion 2.1 base. https://huggingface.co/stabilityai/stable-diffusion-2-1-base, 2022. Accessed: March 5, 2025. 6

[2] Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. *CVPR Workshop on Efficient Deep Learning for Computer Vision*, 2023. 3

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster, 2023. 3

[4] Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. Masactrl: Tuning-free mutual self-attention control for consistent image synthesis and editing, 2023. 2, 7

[5] CompVis. Stable diffusion v1.4. https://huggingface.co/CompVis/stable-diffusion-v1-4, 2022. Accessed: March 5, 2025. 6

[6] Y. Dalva, K. Venkatesh, and P. Yanardag. Fluxspace: Disentangled semantic editing in rectified flow transformers. 2024. 2, 3

[7] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. 3

[8] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024. 2

[9] K. Feng, Y. Ma, B. Wang, et al. Dit4edit: Diffusion transformer for image editing. 2024. 3, 7

[10] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion, 2022. 3

[11] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or. Prompt-to-prompt image editing with cross attention control. 2022. 2, 4, 6, 7

[12] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*, 2020. 4

[13] Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Pnp inversion: Boosting diffusion-based editing with 3 lines of code. *International Conference on Learning Representations (ICLR)*, 2024. 2, 3, 7

[14] Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. *arXiv preprint arXiv:2412.08629*, 2024. 3

[15] Black Forest Labs. Flux.1-dev. https://huggingface.co/black-forest-labs/FLUX.1-dev, 2024. Accessed: March 5, 2025. 6

[16] S. Lin, B. Liu, J. Li, and X. Yang. Common diffusion noise schedules and sample steps are flawed. 2024. 2

[17] Y. Lipman, R.T.Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2023. 4

[18] H. Liu, W. Zhang, J. Xie, et al. Faster diffusion via temporal attention decomposition. 2024. 3

[19] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022. 3

[20] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022.

[21] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2023. 3

[22] Jingyi Lu, Xinghui Li, and Kai Han. Regiondrag: Fast region-based image editing with diffusion models, 2024. 3, 7

[23] Shilin Lu, Yanzhu Liu, and Adams Wai-Kin Kong. Tf-icon: Diffusion-based training-free cross-domain image composition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2294–2305, 2023. 7

[24] Shilin Lu, Yanzhu Liu, and Adams Wai-Kin Kong. Tf-icon: Diffusion-based training-free cross-domain image composition, 2023. 2, 3, 4, 7

[25] X. Ma, G. Fang, and X. Wang. Deepcache: Accelerating diffusion models for free. *arXiv preprint arXiv:2312.00858*, 2023. 3

[26] X. Ma, G. Fang, M.B. Mi, and X. Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *arXiv preprint arXiv:2406.01733*, 2024. 3

[27] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or. Null-text inversion for editing real images using guided diffusion models. 2022. 2

[28] Shen Nie, Hanzhong Allan Guo, Cheng Lu, Yuhao Zhou, Chenyu Zheng, and Chongxuan Li. The blessing of randomness: Sde beats ode in general diffusion-based image editing, 2024. 3

[29] Z. Pan, R. Gherardi, X. Xie, and S. Huang. Effective real image editing with accelerated iterative diffusion inversion. 2023. 2

[30] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 2

[31] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023. 3

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 6

[33] L. Rout, Y. Chen, N. Ruiz, C. Caramanis, S. Shakkottai, and W.-S. Chu. Semantic image inversion and editing using rectified stochastic differential equations. *arXiv preprint*, arXiv:2410.10792, 2024. 2, 6

[34] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation, 2023. 3

[35] P. Selvaraju, T. Ding, T. Chen, I. Zharkov, and L. Liang. Fora: Fast-forward caching in diffusion transformer acceleration. *arXiv preprint arXiv:2407.01425*, 2024. 3, 2

[36] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. *arXiv preprint arXiv:2306.14435*, 2023. 3, 7

[37] SimianLuo. Lcm dreamshaper v7. https://huggingface.co/SimianLuo/LCM_Dreamshaper_v7, 2024. Accessed: March 5, 2025. 6

[38] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. 2022. 2, 3, 6

[39] PIE-Bench Team. Pie-bench: Prompt-driven image editing benchmark. https://forms.gle/hVMkTABb4uvZVjme9, 2024. Accessed: March 5, 2025. 6

[40] Visual AI Team. Regiondrag: Interactive region-based image editing. https://github.com/Visual-AI/RegionDrag, 2024. Accessed: March 5, 2025. 7

[41] J. Wang, J. Pu, Z. Qi, et al. Taming rectified flow for inversion and editing. 2024. 2

[42] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans, 2018. 3

[43] Y. Wang, W. Zhang, J. Zheng, and C. Jin. Primecomposer: Faster progressively combined diffusion for image composition with attention steering. 2024. 3, 7

[44] E. Xie, J. Chen, J. Chen, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. 2024. 2, 3

[45] S. Xu, Y. Huang, J. Pan, Z. Ma, and J. Chai. Inversion-free image editing with natural language. 2023. 2, 7

[46] Ben Xue, Shenghui Ran, Quan Chen, Rongfei Jia, Binqiang Zhao, and Xing Tang. Dccf: Deep comprehensible color filter learning framework for high-resolution image harmonization, 2022. 3, 7

[47] Evelyn Zhang, Jiayi Tang, Xuefei Ning, and Linfeng Zhang. Training-free and hardware-friendly acceleration for diffusion models via similarity-based token pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. 3

[48] Xuanjia Zhao, Jian Guan, Congyi Fan, Dongli Xu, Youtian Lin, Haiwei Pan, and Pengming Feng. Fastdrag: Manipulate anything in one step. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 3, 7

[49] Tianrui Zhu, Shiyi Zhang, Jiawei Shao, and Yansong Tang. Kv-edit: Training-free image editing for precise background preservation. *arXiv preprint arXiv:2502.17363*, 2025. 3, 4

[50] C. Zou, X. Liu, T. Liu, S. Huang, and L. Zhang. Accelerating diffusion transformers with token-wise feature caching. *arXiv preprint arXiv:2410.05317*, 2024. 3, 2

[51] C. Zou, E. Zhang, R. Guo, et al. Accelerating diffusion transformers with dual feature caching. *arXiv preprint arXiv:2412.18911*, 2024. 3, 2