A REIMPLEMENTATION OF SWIFTEDIT FOR ONE-STEP TEXT-GUIDED IMAGE EDITING

ANDREW CHANG [ANDREW08@SEAS], MATTHEW KUO [MKUO@SEAS], MEGAN YANG [MEGYANG@SEAS], DAVID ZHAN[DAZHAN@SEAS],

ABSTRACT. We present a reimplementation and empirical analysis of SwiftEdit, a state-of-the-art one-step text-guided image editing framework based on distilled diffusion models. Our work reproduces the full SwiftEdit inference pipeline, including one-step inversion, self-guided mask extraction, and mask-aware attention rescaling, and evaluates its behavior on the PIE-Bench dataset. We further introduce two training-free modifications: an edge-aware refinement to reduce boundary spill in editing masks and an adaptive attention rescaling heuristic based on mask confidence and uncertainty. Quantitative results show that while naive alternative mask extraction strategies degrade background preservation and edit localization, our refinements significantly reduce boundary artifacts and unintended background changes. However, these improvements come with a trade-off in edited-region semantic alignment, highlighting a fundamental tension between localization accuracy and edit strength. Overall, our findings emphasize the critical role of reliable mask estimation in enabling effective one-step text-guided image editing.

1. INTRODUCTION

Text guided image editing enables users to modify images to natural language instructions, leveraging the priors learned by large-scale text-to-image diffusion models. However, most existing implementations use computationally expensive multi-step diffusion processes requiring 50+ denoising steps, which is prohibitive for interactive applications requiring real-time feedback

Recent developments have demonstrated that high-quality images can be generated in a single forward pass through knowledge distillation from multi-step teacher models (e.g., SwiftBrushV2). This raises the question: can one-step diffusion models enable similarly fast text-guided image editing?

The goal of this project is to present a re-implementation, evaluation, and modifications to SwiftEdit, which is a state of the art text-guided image editor. At its core, SwiftEdit speeds up existing methods by doing inversion and editing in a single step each, which yields a remarkable inference time of 0.23 seconds.

1.1. Contributions.

- **Reimplementation:** We reproduced the full SwiftEdit inference pipeline (one-step inversion, self-guided mask extraction, and mask-aware ARaM generation) from the official repository.
- **Mask refinement without retraining:** We implemented an *edge-aware* refinement to the self-guided mask to reduce boundary spill and unintended background drift.
- **Adaptive ARaM scaling:** We introduced a training-free heuristic that adapts foreground/background ARaM scales from a distribution-based confidence score and mask uncertainty (entropy), improving robustness when masks are ambiguous.
- **Empirical findings:** On PieBench samples, our full modifications slightly improved PSNR/MSE and strongly reduced edge-focused artifact ratios and spill, but sometimes reduced edited-region CLIP alignment, revealing a consistent localization vs. edit-strength trade-off.

2. BACKGROUND

2.1. **Diffusion-Based Text-to-Image Models.** Modern text-to-image generation systems are commonly built on diffusion models, which synthesize images by iteratively denoising a latent variable conditioned on a text prompt. Given an input noise sample and a text embedding, a diffusion model predicts the noise component at each step and gradually refines the latent into an image representation. While this process produces high-quality results, it typically requires tens of denoising steps, resulting in high inference latency.

2.2. **One-Step Diffusion and Distillation.** Recent work has shown that diffusion models can be accelerated through distillation, enabling high-quality image generation in a single forward pass. One-step diffusion models train a student generator to directly map noise and text embeddings to an image latent, amortizing the iterative denoising process into a single network evaluation. SwiftEdit builds on this idea by adopting SwiftBrushV2, a one-step text-to-image diffusion model, as its generative backbone.

2.3. **Image-Prompt Adapter.** To preserve source image content during editing, SwiftEdit incorporates the Image-Prompt Adapter (IP-Adapter), which injects image features into the diffusion model through a decoupled cross-attention mechanism. This design allows the model to balance text-guided edits with visual fidelity to the source image by separately controlling text and image conditioning strengths. IP-Adapter plays a central role in SwiftEdit by enabling identity preservation and background consistency during one-step image editing.

3. Related Work

Text-guided image editing methods typically map images to noisy latents via inversion, then denoise under a new prompt [7]. Localization remains challenging: attention-control methods [5] guide edits spatially, while IP-Adapter-style approaches [6] inject image features to preserve identity. These methods are flexible but slow, requiring many denoising steps. SwiftEdit [1] addresses latency by combining one-step inversion with a one-step generator (SwiftBrushV2 [2,3]), achieving end-to-end editing in two forward passes. Its ARaM mechanism uses mask-guided attention rescaling to localize edits. However, mask errors propagate into boundary artifacts and background drift. We evaluate on PIE-Bench [4], which provides paired prompts and ground-truth masks, and introduce training-free refinements targeting these failure modes.

4. Approach

4.1. **Overview.** SwiftEdit performs one-step inversion (4.2), extracts a self-guided mask (4.3), and generates edited images via mask-aware attention rescaling (4.4). Our implementation follows this architecture.

4.2. **One-Step Inversion Network.** The inversion network F_θ shares the same UNet architecture as the SwiftBrushv2 generator and is initialized with its weights. Given an image latent z and text embedding c_y, it predicts the inverted noise $\hat{\epsilon} = F_\theta(z, c_y)$ that would reconstruct z when passed through the generator. A key design choice is pairing the inversion network with the IP-Adapter-enhanced generator GIP (Section 2.3) during training, so the image-conditioning branch reduces the burden on the inverted noise to encode fine-grained visual details.

4.3. **Self-Guided Mask Extraction.** To identify editing regions without user input, SwiftEdit exploits the inversion network's sensitivity to prompt changes. The mask is computed by inverting the source image with both the source and edit prompts at an intermediate timestep (default $t = 500$), then measuring the difference: $M = \text{normalize}(|\hat{\epsilon}^{source} - \hat{\epsilon}^{edit}|)$. Specifically, the difference is averaged across channels, clamped to $[0, \mu \cdot r]$ where μ is the mean and r is a clamping rate (default 3.0), normalized, and binarized with threshold 0.5. The implementation in editing_utils.py follows this procedure directly.

4.4. **Attention Rescaling for Mask-Aware Editing (ARaM).** During generation with G^{IP}, standard decoupled cross-attention applies a global image condition scale s_x uniformly across the image. ARaM modifies this to use region-specific scales guided by the editing mask M. The attention output becomes:

$$h_l = s_y \cdot M \cdot \text{Attn}(Q_l, K_y, V_y) + s_{edit} \cdot M \cdot \text{Attn}(Q_l, K_x, V_x) + s_{non-edit} \cdot (1 - M) \cdot \text{Attn}(Q_l, K_x, V_x) \quad (1)$$

where s_y controls text-image alignment strength in the editing region, s_{edit} controls image conditioning in the foreground (editing region), and $s_{non-edit}$ controls image conditioning in the background. By setting $s_{edit} = 0$ and $s_{non-edit} = 1$, the edited region relies purely on the text prompt while the background preserves source image fidelity through strong image conditioning. The parameter s_y (default 2.0) provides additional control over editing strength. The implementation in mask_controller.py and mask_attention_processor.py realizes this mechanism through specialized attention processors that compute separate attention maps for foreground and background regions, then blend them according to the mask.

4.5. **Modification 1: Cross-Attention-Based Editing Mask Extraction. Note:** This attention-based mask extraction was explored conceptually and implemented on top of our SwiftEdit reimplementation, but was not included in the PIE-Bench quantitative evaluation. All reported quantitative results evaluate only the training-free mask refinement and adaptive ARaM scaling described below.

SwiftEdit's core mechanism relies on self-guided masks derived from differences in noise predictions under the source and edit prompts. As an alternative, we propose an attention-based mask extraction strategy that leverages the generator's text cross-attention patterns. Because cross-attention explicitly encodes spatial–textual alignment, differences in attention between the source and edit prompts can provide a semantically grounded estimate of the intended edit region.

Concretely, during the one-step forward pass, we record cross-attention maps from the generator under both the source and edit prompts, denoted as A^{source} and A^{edit}, using the same noisy latent z. We extract attention only from the mid and up blocks, which offer higher semantic and spatial resolution, based on qualitative validation. The resulting attention differences are then aggregated to produce an editing mask.

Next, we want to find the absolute difference between the attention maps for each layer using the following equation:

$$\Delta_\ell = \left| A_\ell^{source} - A_\ell^{edit} \right|.$$

We then average across heads and text tokens to obtain a spatial importance map:

$$d_\ell(i) = \frac{1}{H_\ell T} \sum_{h=1}^{H_\ell} \sum_{t=1}^{T} \Delta_\ell(h, i, t), \quad i \in \{1, \ldots, N_\ell\}.$$

Each d_ℓ is reshaped to its spatial grid, resized to a common resolution, and combined across layers using increasing weights for later layers:

$$d(i) = \sum_\ell w_\ell \, d_\ell(i), \qquad \sum_\ell w_\ell = 1.$$

Finally, following the SwiftEdit paper, we clamp and normalize the aggregated map using the follow equation:

$$\tilde{d} = \frac{\text{clip}(d,\ 0,\ \mu \cdot r)}{\mu \cdot r}, \quad \mu = \mathbb{E}[d],\ r = 3.0,$$

and threshold to obtain a binary editing mask:

$$M_{\text{attn}}(i) = \mathbb{I}[\tilde{d}(i) > 0.5].$$

The resulting mask is resized to the latent resolution and passed into ARaM, replacing the original self-guided mask when enabled.

We have the following results.

Method	**PSNR$_{\text{BG}}$** ↑	**MSE$_{\text{BG}}$** ↓	**CLIP-W** ↑	**CLIP-E** ↑	**Time (s)** ↓
SwiftEdit (Baseline)	**23.33**	**6.60**	25.16	21.25	0.23
Noise-Based (our impl)	17.68	19.26	**76.70**	91.10	**0.22**
Attention-Based (mod 1)	17.58	19.48	76.42	**98.39**	0.23

TABLE 1. Average performance on PIE-Bench comparing SwiftEdit baseline with noise-based and attention-based mask extraction methods. Noise and attention results are averaged over 140 samples per method.

4.6. **Interpretation of results.** Compared to the SwiftEdit baseline, both the noise-based and attention-based mask variants exhibit noticeably lower background fidelity, as reflected by reduced PSNR$_{\text{BG}}$ and higher MSE$_{\text{BG}}$. This indicates that the inferred masks are less precise, causing a larger portion of the background to be unintentionally modified during editing. At the same time, both variants achieve substantially higher CLIP-W and CLIP-E scores, suggesting that the edited images remain globally and locally more similar to the source image. This behavior implies that the edits are more conservative and less localized: imprecise masks dilute the intended edit by spreading changes across the image while preserving much of the original content, leading to higher semantic similarity but weaker targeted modifications. Overall, these results highlight the importance of accurate mask estimation in SwiftEdit-style pipelines, as mask errors directly trade off edit strength for background preservation and semantic consistency.

FIGURE 1. Example SwiftEdit baseline edit ("make the cake square").

4.7. **Modification 2: Edge-Aware Mask Refinement (Boundary Spill Reduction).** SwiftEdit's self-guided mask can exhibit *boundary spill*, where uncertain edge regions allow unintended background edits. We implement a **training-free refinement** that down-weights mask values at boundaries when the difference signal δ (channel-averaged $|\hat{\epsilon}^{src} - \hat{\epsilon}^{edit}|$) exhibits strong edges.

We compute a boundary band $b = \mathrm{dilate}(M) - \mathrm{erode}(M)$ and an edge-strength map e via Laplacian operator on δ, normalized to $[0, 1]$. The mask is linearly attenuated at boundaries:

$$\tilde{M}(i) = M(i)\big(1 - \beta\, e(i)\, b(i)\big),$$

where β controls edge suppression strength. We then blend with the original mask:

$$M'(i) = \alpha\, M(i) + (1 - \alpha)\, \tilde{M}(i),$$

using preservation coefficient α. This targets **Edge-Ratio** and **Spill** improvements, with potential trade-offs in **CLIP-E** if the mask becomes more conservative.

4.8. **Modification 3: Adaptive ARaM Scaling via Confidence + Mask Uncertainty.** SwiftEdit's ARaM uses fixed scaling factors for foreground/background regions, which can be suboptimal when masks are noisy or uncertain. We implement a **training-free adaptive scaling heuristic** that adjusts ARaM scales based on two inference-time signals: **(i) distribution confidence** c_{dist} (how separable the difference distribution is) and **(ii) mask uncertainty** u_{mask} (Bernoulli entropy of the soft mask).

We normalize c_{dist} to $[0, 1]$ via $\bar{c}_{\mathrm{dist}} = \mathrm{clip}(c_{\mathrm{dist}}/c_0, 0, 1)$ where $c_0 = 2.0$, and compute mask entropy:

$$u_{\mathrm{mask}} = \frac{1}{HW} \sum_i \Big(-M(i) \log M(i) - (1 - M(i)) \log(1 - M(i))\Big),$$

normalized by $\log 2$ to obtain $\bar{u}_{\mathrm{mask}} \in [0, 1]$. Combined confidence is:

$$c_{\mathrm{comb}} = 0.6\, \bar{c}_{\mathrm{dist}} + 0.4\, \big(1 - \bar{u}_{\mathrm{mask}}\big).$$

We map c_{comb} to ARaM scales via a monotone rule: lower confidence produces more conservative edits (higher background preservation, lower foreground modification), while higher confidence enables stronger edits. This targets **Spill** and **BG-Change** improvements, with potential trade-offs in **CLIP-E** due to the localization vs. edit-strength trade-off.

5. EXPERIMENTAL RESULTS

5.1. **Models and Architecture.** We use the **released SwiftEdit checkpoints** from the official repository: the one-step inversion network F_θ, the SwiftBrushV2 one-step generator G, and the IP-Adapter conditioning branch. We **do not retrain** any model weights. Instead, we evaluate **inference-only** changes that modify (i) post-processing of the self-guided mask and (ii) run-time selection of ARaM scaling parameters. All results are reported *relative to the unmodified SwiftEdit baseline* under identical pretrained weights.

Config	PSNR↑	MSE↓	CLIP-W↑	CLIP-E↑
Baseline	5.52	183179248.05	23.53	23.78
Edge Only	5.56	181729664.71	23.44	21.01
Scales Only	5.51	183736057.94	23.46	20.17
All Improvements	5.55	181860582.68	23.59	21.02

TABLE 2. Average PIE-Bench results using paper-style metrics.

Config	Mask-Q↑	Edge-Blur↓	Edge-Pres↑	Edge-Ratio↓	Spill↓	BG-Change↓
Baseline	0.272	0.0590	0.534	0.0364	0.841	0.2240
Edge Only	0.274	0.0609	0.528	0.0060	0.778	0.2225
Scales Only	0.274	0.0617	0.528	0.0061	0.783	0.2242
All Improvements	0.274	0.0609	0.527	0.0060	0.779	0.2226

TABLE 3. Average PIE-Bench results using custom locality/boundary metrics.

5.2. **Dataset, Evaluation Protocol, and Metrics.** We evaluate our methods on **PIE-Bench** (HuggingFace), which provides paired source images and prompts, target prompts, and evaluation metadata including reference masks. For each sample, we evaluate four configurations under identical hyperparameters: **Baseline**, **Edge Only** (edge-aware mask refinement), **Scales Only** (adaptive ARaM scaling), and **All Improvements** (both). All images are resized to 512×512 to match the pretrained VAE/UNet setup, following SwiftEdit preprocessing: normalization to $[-1, 1]$, VAE encoding, latent-space editing, and decoding back to pixel space.

We report two classes of metrics. First, we use **paper-style metrics** from SwiftEdit: PSNR and MSE to measure background preservation, and CLIP similarity to assess semantic alignment for the whole image (CLIP-W) and edited regions (CLIP-E). Second, we introduce **locality-focused diagnostics** to explicitly measure our target failure modes, including Edge-Ratio, Edge-Blur, and Edge-Pres for boundary behavior, as well as Spill and BG-Change to quantify unintended background modification.

5.3. **Interpretation of Results.** Overall, our modifications primarily improve **localization robustness** rather than producing visually dramatic changes. The clearest gain is a large reduction in boundary-focused artifacts: **Edge-Ratio** drops by $\sim 6.1\times$ ($0.0364 \rightarrow 0.0060$), and **Spill** decreases ($0.841 \rightarrow 0.779$), indicating less unintended background change. PSNR/MSE improve slightly, consistent with modestly better preservation.

The main downside is reduced edited-region alignment: **CLIP-E** decreases for all modified variants, suggesting a localization vs. edit-strength trade-off. In practice, the improved variants often look similar to baseline when baseline already localizes well; the benefit shows up most on harder cases as cleaner boundaries and reduced drift.

6. DISCUSSION

Our results show that SwiftEdit's performance is highly sensitive to the quality of the inferred editing mask. While alternative mask extraction strategies based on noise differences and cross-attention patterns are conceptually appealing, their imprecision leads to a clear degradation in background preservation, as evidenced by lower PSNR_{BG} and higher MSE_{BG}. At the same time, these variants exhibit higher CLIP-W and CLIP-E scores, indicating that edits remain semantically closer to the source image but are weaker and less localized. This highlights a fundamental trade-off in SwiftEdit-style pipelines: inaccurate masks tend to dilute the intended edit by spreading changes across the image, preserving global semantics at the cost of edit strength and localization.

Our training-free refinements—edge-aware mask attenuation and adaptive ARaM scaling—partially mitigate these issues by reducing boundary spill and unintended background drift without modifying model weights. However, the accompanying drop in edited-region CLIP alignment suggests that improving localization often comes at the expense of edit intensity. Overall, these findings underscore the central role of reliable mask estimation in one-step text-guided image editing and suggest that future improvements may require either stronger mask supervision or lightweight fine-tuning to better balance localization and semantic alignment.

REFERENCES

[1] D. T. Nguyen, M. Vilela, D. H. Tran, A. Tavakoli, and B. Schneemann. SwiftEdit: Lightning-Fast Text-Guided Image Editing via One-Step Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. https://openaccess.thecvf.com/content/CVPR2025/papers/Nguyen_SwiftEdit_Lightning_Fast_Text-Guided_Image_Editing_via_One-Step_Diffusion_CVPR_2025_paper.pdf

[2] D. T. Nguyen, M. Vilela, D. H. Tran, A. Tavakoli, and B. Schneemann. SwiftBrush v2: One-Step Text-to-Image Diffusion Model with Variational Score Distillation. arXiv preprint arXiv:2503.04726, 2025.

[3] Y. Qian, D. Li, B. Ghanem, and M. Elhoseiny. SwiftBrush: One-Step Text-to-Image Diffusion Model with Variational Score Distillation. arXiv preprint arXiv:2501.09016, 2025.

[4] X. Ju, A. Zeng, Y. Bian, S. Liu, and Q. Xu. Direct Inversion: Boosting Diffusion-based Editing with 3 Lines of Code. arXiv preprint arXiv:2310.01506, 2023. (Introduces PIE-Bench.)

[5] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or. Prompt-to-Prompt Image Editing with Cross Attention Control. arXiv preprint arXiv:2208.01626, 2022.

[6] H. Ye, J. Huang, S. Liu, and others. IP-Adapter: Text Compatible Image Prompt Adapter for Text-to-Image Diffusion Models. arXiv preprint arXiv:2308.06721, 2023.

[7] T. Brooks, A. Holynski, and A. A. Efros. InstructPix2Pix: Learning to Follow Image Editing Instructions. arXiv preprint arXiv:2211.09800, 2022.

[8] N. Tumanyan, A. Voynov, S. Bagon, and T. Dekel. PnP Inversion: Boosting Diffusion-Based Edit Models with Prompt-to-Prompt Guidance. In *International Conference on Learning Representations (ICLR)*, 2024.