Prompt-Guided Image Editing with Masked Logit Nudging in Visual Autoregressive Models

Supplementary Material

Supplementary Material

This supplementary document provides additional analysis and implementation details for *Masked Logit Nudging* (MLN). In particular, we include:

1. Detailed analysis of the cross-attention–driven edit masks, including quantitative mask–GT comparisons, threshold sensitivity, and layer/head ablations (Sec. 6.1.2).
2. Additional comparison and ablations of nudging schedule (Sec. 6.2).
3. Further MLN ablations and hyperparameters (Sec. 6.3).
4. Extended analysis of quantization errors and the proposed quantization refinement procedure (Secs. 6.4).
5. Details and qualitative samples of the reconstruction experiments (Sec. 6.5).
6. Details and additional qualitative samples of the editing experiments (Sec. 6.6).
7. Adapted upscaled PIE-benchmark at 1024px (Sec. 6.7).
8. Recaptioning for reconstruction experiments at 1024px (Sec. 6.8).
9. Additional qualitative editing samples (Sec. 6.9).
10. More ablations (Sec. 6.10).
11. Failure Analysis (Sec. 6.11).

6.1. Cross-attention mask analysis

Our masking mechanism follows the attention-based editing philosophy of DDIM inversion and P2P [14], but applies it directly to the cross-attention activations of the VAR transformer, which uses the same multi-head attention structure as GPT-style models. To extract these activations, we run two short regeneration passes—one with the source prompt t_s and one with the target prompt t_t—from the high-resolution scales (s=9 for 512 px and s=13 for 1024 px). The difference between these attention maps yields a spatial relevance map, which we threshold to obtain the edit mask used by MLN.

In the following we analyze this masking process in detail, focusing on:

- how mask-related hyperparameters (regeneration latency, percentile threshold q, layer/head selection) affect mask quality and editing performance.
- how the masks align with the PIE ground-truth edit regions.

Unless otherwise noted, all statistics are computed on the PIE-Benchmark for 512 px(PIE-512) resolution using the SWITTI backbone [43].

6.1.1. Hyperparameter and latency

Mask related regeneration latency. To extract cross-attention maps, we run regeneration from $s_{\mathbf{M}}$ and record the attention tensors $\mathbf{A}^s$ (source prompt) and $\mathbf{A}^t$ (target prompt). The latency below reflects the total time required to compute both $\mathbf{A}^s$ and $\mathbf{A}^t$ for a single image. We benchmark this trade-off on PIE-512 for $s_{\mathbf{M}} \in \{5, 6, 7, 8, 9\}$.

Table 4. **Latency and precision for varying s (512 px).**

$s_{\mathbf{M}}$	Latency (ms)↓	Precision (%)↑
5	325	63
6	122	67
7	80	69
8	43	68
9*	20	71

Latency decreases for larger $s_{\mathbf{M}}$ because fewer scale predictions are executed: when $s_{\mathbf{M}} = 6$, the model still processes four additional scales, each requiring a full autoregressive forward pass over increasingly large token grids. Although later scales contain more tokens, the dominant cost arises from the repeated multi-scale predictions at earlier stages(since they are sequential and not parallelizable), making shallow $s_{\mathbf{M}}$ values substantially slower overall.

Mask precision increases steadily with higher regeneration scale $s_{\mathbf{M}}$ and peaks near $s_{\mathbf{M}}$=9, which corresponds to almost the full latent resolution (K=10). Based on this trade-off, we adopt $s_{\mathbf{M}}$=9 for 512 px (and $s_{\mathbf{M}}$=13 for 1024 px) in all subsequent experiments.

Threshold sensitivity. The binary mask $\mathbf{M}$ is obtained by selecting the top-q percentile of cross-attention differences, making q the main control over mask sparsity. Low q yields overly small masks, while high q produces masks that spill into the background.

We evaluate $q \in \{60, 70, 80, 90\}$ on PIE-512 and measure mask coverage, IoU with the ground-truth edit region, and MLN editing quality.

Table 5. **Effect of threshold q on mask sparsity and editing quality (PIE-512).**

q	Coverage (%)↓	IoU (%)↑	PSNR↑	LPIPS↓	CLIP↑
60	4.8	51	29.1	0.128	0.322
70	7.3	57	29.4	0.121	0.331
80*	10.6	63	29.7	0.118	0.339
90	18.2	61	29.0	0.132	0.336

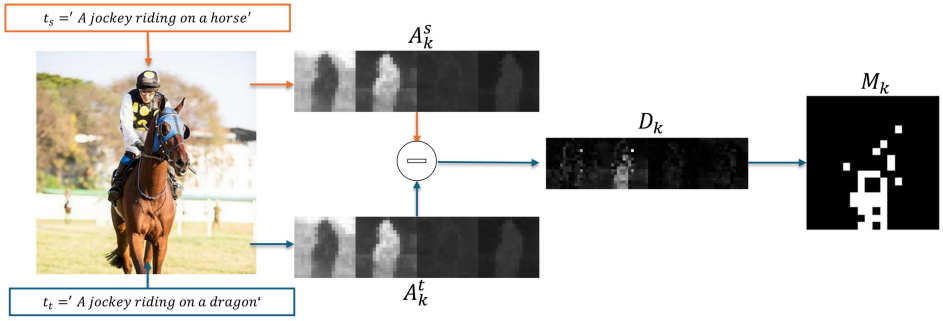

Figure 5. **Mask construction overview.** Cross-attention differences between source and target prompts identify editable regions.

Overall, q=80 offers the best trade-off: it yields the highest IoU and strong editing performance without unnecessary background changes. We adopt $q = 80$ for 512px and $q = 63$ for 1024px.

Layer and head ablations. We aggregate cross-attention maps by averaging over all heads (as also done in Prompt-to-Prompt [14]) and study which transformer decoder blocks provide the strongest and most stable attention differences. Visually, we observe that useful attention structure emerges only from layers 3–27: early blocks (0–2) produce noisy activations, while late blocks (28–29) are overly localized and inconsistent. The middle layers capture both spatial layout and fine-grained attribute changes.

To quantify this, we compute masks from different layer ranges on PIE-512 (q=80) and measure IoU with ground-truth edit regions together with MLN editing performance.

Table 6. **Layer-range ablation (PIE-512, q=80).** Middle blocks yield the most coherent masks and best editing fidelity.

Layer Range	IoU (%)↑	PSNR↑	LPIPS↓
Early (0–2)	41	28.6	0.151
Middle (3–27)*	63	29.7	0.118
Late (28–29)	52	29.2	0.134
All (0–29)	61	29.6	0.123

Figure 6 shows the attention-difference maps for all 30 blocks, illustrating that layers 3–27 provide the cleanest, most semantically aligned masks. Accordingly, we use blocks 3–27 as the default range in all experiments.

6.1.2. Mask vs. Ground-Truth Edit Regions

We compare our cross-attention–derived masks to the ground-truth edit regions on PIE-512. While MLN supports explicit masking, it is important to note that logit nudging alone already maintains much of the background structure. Because the nudging term pulls logits toward the source tokens, the model does not overwrite large regions as aggressively as plain regeneration (can also be seen in fig. 2).

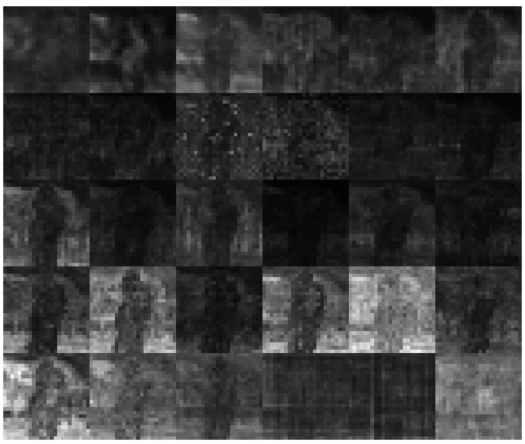

Figure 6. **Cross-attention maps** for all 30 transformer blocks conditioned on the image in fig. 5. Only layers 3–27 yield stable and meaningful masks. Counted left-right from top-bottom.

However, without masking(therefore also without quantization refinement (QR)), the background reconstruction is still worse.

To demonstrate the importance of masking, we compare:

1. logit nudging without a mask – no QR.
2. masked regeneration – no QR.
3. MLN – with QR.

We measure mask IoU against the PIE ground-truth region and report background fidelity.

Table 7. **Mask–GT agreement and background fidelity (PIE-512).** MLN achieves the strongest localization and background preservation.

Method	Mask IoU (%)↑	PSNR (bg)↑	LPIPS (bg)↓	CLIP (edit)↑
Logit nudging	–	25.8	85.2	24.4
Masked regeneration	57	26.5	79.7	22.2
MLN (ours)	63	29.7	36.5	22.72

Logit nudging without a mask performs well on edit

alignment but fails to preserve background details, confirming that spatial constraints are essential for stable reconstructions. Masked regeneration does not improve background significantly.

Finally, we conclude that applying the mask is beneficial not only for localizing the edit, but also for improving reconstruction outside the mask with the proposed QR.

6.2. Nudging schedules

Masked Logit Nudging applies a scale-dependent guidance weight α_k at each VAR scale k. For 512 px images, SWITTI uses $K{=}10$ scales. We found that the trade-off between edit strength and reconstruction fidelity is best when:

- regeneration from $s{=}6$, and
- nudging is applied from scale $k \geq 7$, with a decreasing schedule toward the finest scales.

In practice, we use schedules that keep α_k high on early editing scales (although these scales are not used during MLN, due to regeneration from $s = 6$) and then gradually reduce it at high-resolution scales (to allow fine details without overshooting). Figure 7 illustrates two representative schedules for 512 px.

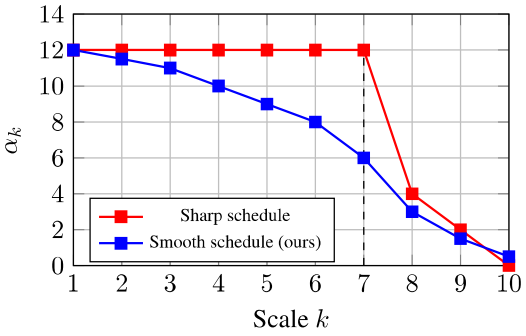

Figure 7. **Nudging schedules at 512 px.** Both schedules use a cutoff at $k_{\text{cut}}{=}7$ (vertical dashed line). We adopt the smooth schedule in all experiments.

In all reconstruction experiments, we keep the same regeneration scale $s{=}6$. In our experiments we use the smooth schedule.

Nudging cutoff k Additionally we ablate different cutoff scales k_{cut} on PIE-512 using the smooth schedule (see Tab. 8). Evaluations include background PSNR, background LPIPS, and CLIP alignment in the edited region. Visual samples are shown in fig. 8.

$k_{\text{cut}}{=}7$ provides the best trade-off between background fidelity (highest PSNR, lowest LPIPS) and edit strength (highest CLIP). Later cutoffs over-constrain fine scales and weaken edits, while earlier cutoffs allow excessive nudging at high resolution and degrade background preservation.

Table 8. **Ablation over cutoff scale k_{cut} (PIE-512, smooth schedule).**

k_{cut}	PSNR (bg)↑	CLIP (edit)↑
5	21.4	26.1
6	22.0	24.4
7*	24.7	22.7
8	25.9	20.5
9	27.9	19.3

6.3. MLN ablations

6.3.1. Component-wise Ablations

We evaluate the contribution of each MLN component on PIE-512. The three components analyzed are:

- **Logit Nudging (LN)** for semantic steering,
- **Cross-Attention Masking (Mask)** for spatial localization, and
- **Quantization Refinement (QR)** for restoring background regions.

Table 9. **Component-wise ablations on PIE-512.** Columns indicate which components are enabled in each variant.

LN	Mask	QR	PSNR (bg)↑	LPIPS (bg)↓	CLIP (edit)↑
✓			23.4	134.2	25.20
	✓		24.5	120.7	21.20
✓	✓		27.6	102.3	22.32
✓	✓	✓	29.70	36.50	22.72

LN enhances edit strength by providing semantic steering, but without RQ background regions are not preserved as good. Masking alone offers almost no improvements by spatially restricting edits, as shown in sec. 6.1.2 it performs only slightly better than LN. The combination of LN and masking yields the largest performance gains, enabling edits that are both semantically aligned and spatially well-localized, however background preservation still suffers. Incorporating all three components produces the most robust results overall.

Bakground preservation weight β The weight β controls the strength of background preservation during MLN (this can be seen in fig. 9). A larger β penalizes deviations outside the mask, improving reconstruction but potentially weakening the edit if set too high. We vary $\beta \in \{0, 2, 4, \ldots, 16\}$ on PIE-512 and measure PSNR (background region) and CLIP similarity (edit region). PSNR increases steadily up to $\beta{=}12$ and then saturates. CLIP improves until $\beta{=}14$, after which it saturates. We therefore use $\beta{=}12$ as the default value.

Regenereation scale s during MLN MLN begins editing from an intermediate VAR scale s, reusing source tokens for all lower scales and applying nudging only at higher

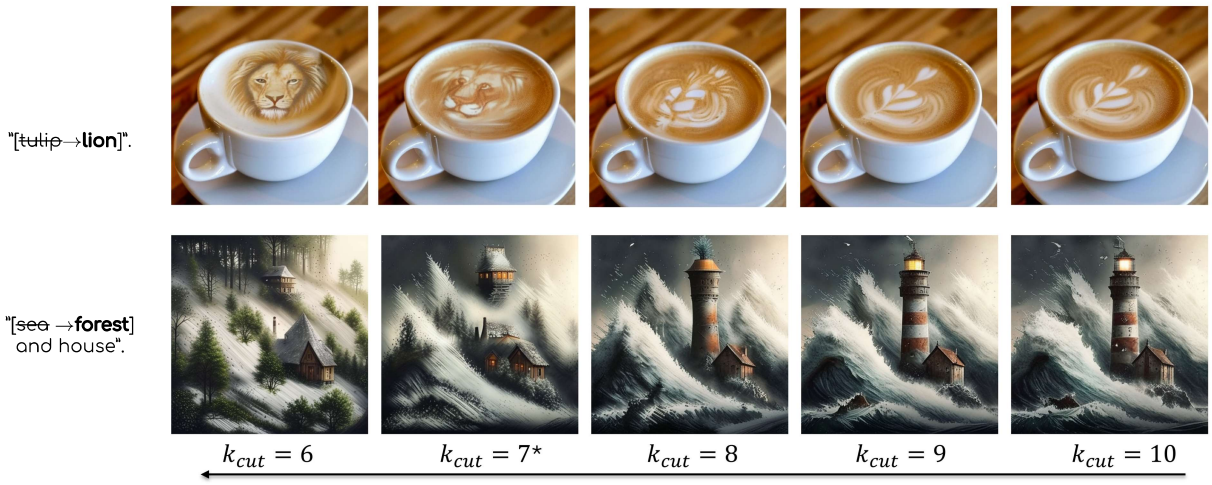

Figure 8. **Nudging cutoff k_{cut}.** Higher k_{cut} preserves more content to the original image (seen at $k_{cut} = 10$). Importantly the upper example utilizes a mask (MLN) to keep edits from the background, while the lower example only uses logit-nudging.

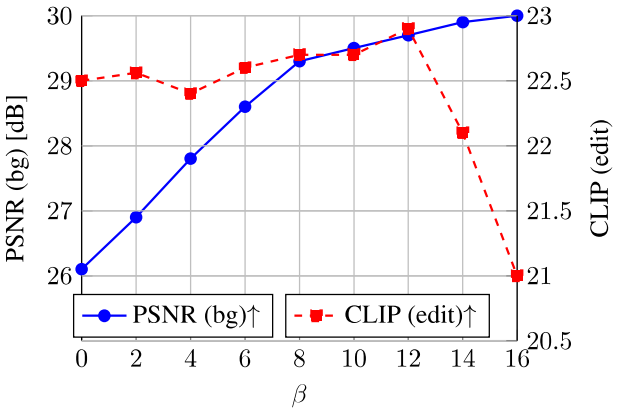

Figure 9. **Ablation over β.** PSNR improves up to β=12. CLIP remains stable until β=14 and then saturates.

scales (see Tab. 10). Choosing s therefore determines the trade-off between preserving global structure and allowing sufficient room for edits to form. We evaluate $s \in \{4, 5, 6, 7, 8\}$ on PIE-512 using identical settings (q=80, β=12).

Table 10. **Ablation of MLN starting scale s (PIE-512)** Increasing s improves background fidelity but weakens edits; s=6 yields the best compromise. Latency measures the time required to run the regeneration/MLN forward pass starting from scale s.

s	PSNR (bg)↑	LPIPS (bg)↓	CLIP (edit)↑	Latency (ms)↓
4	22.1	180.0	23.2	1210
5	26.4	50.3	22.6	962
6*	29.3	36.5	22.7	800
7	30.8	25.2	20.8	413
8	31.0	112.2	18.1	84

Background fidelity improves monotonically with increasing s, while CLIP alignment begins to drop once too few scales remain for meaningful edits. The best overall balance is obtained at s=6, which we adopt as the default for all 512 px experiments (and s=10 for 1024 px).

Sampling Hyperparameters (CFG Schedule) Since early VAR scales are structurally important and later scales contain high-frequency appearance details, we activate CFG-style guidance only in a narrow mid-scale band.

Concretely, we use the following schedule:

- CFG sampling is **enabled starting at scale** $k = 2$, once the global layout has been established.
- CFG sampling is **disabled again at scale** $K - 2$ (i.e., $k = 8$ for K=10 at 512 px).
- Outside this range, we perform standard sampling without CFG adjustment.
- The same schedule is adopted for 1024 px with K=14, i.e., CFG active from k=2 to k=12.

This mid-band CFG improves prompt alignment without destabilizing fine-scale token predictions, and we observe no benefit from applying CFG at the very first or very last scales.

6.4. Extended analysis of quantization errors and quantization refinement

Quantization Errors in VAR. During encoding, each continuous feature map $\mathbf{f}$ is approximated by discrete codebook vectors $\{\mathbf{f}_k\}_{k=1}^{K}$. This introduces a residual error

$$\mathbf{f}_{\text{rest}} = \mathbf{f} - \sum_{k=1}^{K} \mathbf{f}_k,$$

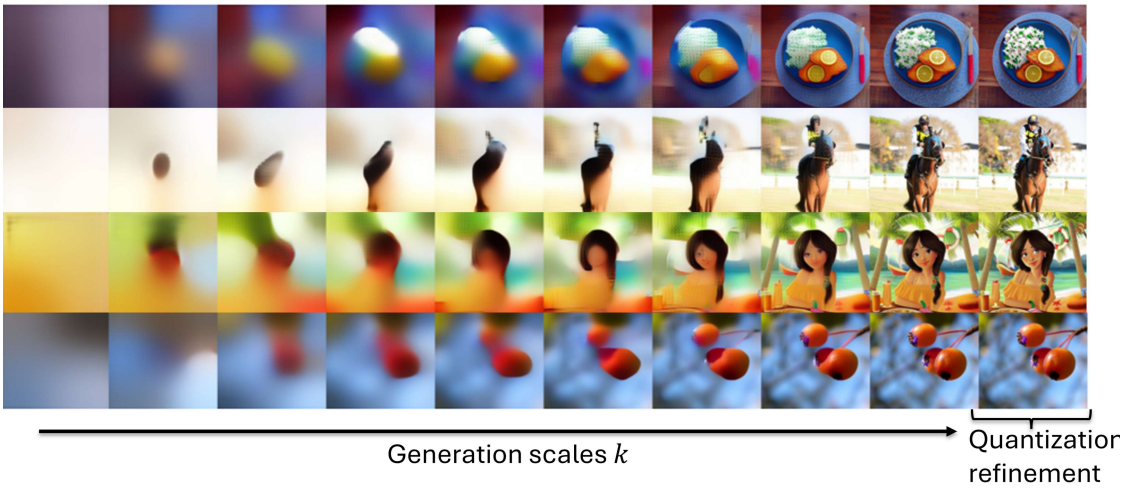

Figure 10. **Generation scales.** Visual comparison of SWITTI generation at all scales k. As k increases, images get more high-frequency details. We choose $k = s = 7$ as regeneration scale.

which accumulates across scales and causes visible distortions after decoding— typically slight color drifts, softened textures, and structural inconsistencies. We visualize these errors in Fig. 11, where the reconstructed image without refinement (fig. 11,top right) deviates from the VQ-manifold, resulting in images that are reconstructed badly.

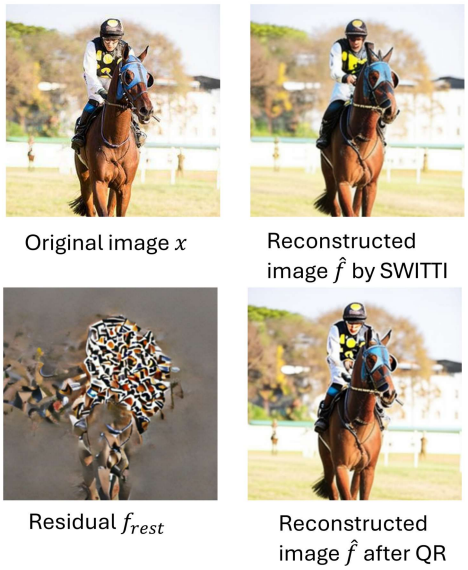

Figure 11. **Visualization of quantization refinement**. Residuals accumulate outside the codebook manifold, causing the default SWITTI reconstruction to make mistakes.

Quantization Refinement - Mathematical Perspective. Because $\mathbf{f}_{\text{rest}}$ generally lies off the codebook manifold spanned by the embeddings $\mathbf{C} = \{c_1, \ldots, c_V\}$, adding it directly to $\hat{\mathbf{f}}$ produces severe artifacts(fig. 11, bottom left). We therefore project the residual back into the codebook space before applying it as a correction.

For each iteration j, we treat the residual $\mathbf{f}_{\text{rest}}^{(j)} \in \mathbb{R}^C$ as continuous observations and compute soft assignment weights

$$w_i^{(j)} = \frac{\exp\left(\langle \mathbf{f}_{\text{rest}}^{(j)}, c_i \rangle / \tau\right)}{\sum_{k=1}^{V} \exp\left(\langle \mathbf{f}_{\text{rest}}^{(j)}, c_k \rangle / \tau\right)},$$

where τ is a temperature controlling assignment sharpness. The projected residual is then

$$\mathbf{f}_{\text{proj}}^{(j)} = \sum_{i=1}^{V} w_i^{(j)} c_i,$$

which lies exactly in the space of the codebook embeddings $\mathbf{c_i}$.

We then update the reconstruction using a step size α,

$$\hat{\mathbf{f}}^{(j+1)} = \hat{\mathbf{f}}^{(j)} + \alpha \mathbf{f}_{\text{proj}}^{(j)},$$

and only apply the correction outside the edit mask $\mathbf{m}$:

$$\hat{\mathbf{f}}^{(j+1)} = \hat{\mathbf{f}}^{(j)} + \alpha \, (\bar{\mathbf{M}}) \odot \mathbf{f}_{\text{proj}}^{(j)}.$$

Iterative refinement The off-manifold residual $\mathbf{f}_{\text{rest}}$ contains components that cannot be removed in a single projection step: each projection eliminates only the portion

Algorithm 1 Quantization refinement

Require: Encoded features $\mathbf{f}$,initial reconstruction $\hat{\mathbf{f}}^{(0)}$, codebook embeddings $\mathbf{C}$, iterations T, temperature τ, step size α, tolerance ε,mask $\mathbf{M}$.

$\hat{\mathbf{f}} \leftarrow \hat{\mathbf{f}}^{(0)}$
$\mathbf{f}_{\text{out}} \leftarrow \hat{\mathbf{f}}^{(0)}$ ▷ *Final refined features*
for $j = 1, 2, \ldots, T$ **do**
  $\mathbf{f}_{\text{rest}} \leftarrow \mathbf{f} - \hat{\mathbf{f}}$ ▷ *Residual*
  $r \leftarrow \|\mathbf{f}_{\text{rest}}\|_2$ (e.g. mean ℓ_2 norm)
  if $r < \varepsilon$ **then**
    break ▷ *Early stopping*
  end if
  Reshape $\mathbf{f}_{\text{rest}}$ to $\mathbf{Z} \in \mathbb{R}^{N \times C}$, where $N = BHW$
  $\mathbf{S} \leftarrow \mathbf{Z}\mathbf{C}^\top \in \mathbb{R}^{N \times V}$ ▷ *Similarities to codebook*
  $\mathbf{W} \leftarrow \mathrm{softmax}(\mathbf{S}/\tau, \mathrm{dim} = V)$ ▷ *Soft assignments*
  $\mathbf{Z}_{\text{proj}} \leftarrow \mathbf{W}\mathbf{C} \in \mathbb{R}^{N \times C}$ ▷ *Projection to codebook space*
  Reshape $\mathbf{Z}_{\text{proj}}$ back to $\mathbf{f}_{\text{rest}}^{\text{proj}} \in \mathbb{R}^{B \times C \times H \times W}$
  $\hat{\mathbf{f}} \leftarrow \hat{\mathbf{f}} + \alpha\, \mathbf{f}_{\text{rest}}^{\text{proj}}$
  $\mathbf{f}_{\text{out}} \leftarrow \mathbf{f}_{\text{out}} + \alpha\, \mathbf{f}_{\text{rest}}^{\text{proj}} \odot (\bar{\mathbf{M}})$ ▷ *Refine outside edit mask*
end for
Output: $\mathbf{f}_{\text{out}}$

explainable by the codebook, while the remaining off-manifold residual changes shape after every update. Thus, repeating the projection–update cycle gradually decreases the residual norm,

$$\|\mathbf{f}_{\text{rest}}^{(j)}\|_2 \downarrow,$$

until the correction becomes negligible, yielding a reconstruction closer to the original $\mathbf{f}$ while keeping the edited region intact. In practice we use $j = 5$ iteration and $\tau = 0.2$ at 512px and $j = 3$ with $\tau = 0.8$ at 1024px. The pseudocode can be seen in algorithm 1.

Latency. The quantization–refinement step is computationally negligible compared to the VAR forward pass. Each iteration involves only matrix multiplications with the codebook ($V \times C$) and per-pixel softmax operations, both of which are highly optimized on modern GPUs. In practice, running 10 refinement iterations adds less than **10** ms of overhead for both 512 and 1024 resolutions, making the procedure effectively free relative to the overall editing pipeline. As a result, the refinement can be applied by default without compromising real-time editing speed.

6.5. Details and qualitative samples of reconstuction experiments

6.5.1. Reconstruction methods

To evaluate reconstruction fidelity in the zero-edit setting (i.e., source and target prompts identical), we benchmark MLN against a set of diffusion-based baselines. This subsection summarizes all methods included in the reconstruction experiments, together with the backbones and sampling configurations used in our evaluation. All experiments are performed on COCO (512 px) and the curated OpenImages subset (1024 px) using the official evaluation splits. In the experiments we always relied on the source code provided by the authors, if not otherwise indicated, whenever the source code was available we tested the provided configurations and chose the best one with respect to reconstruction performance.

Overview of evaluated methods at 512px. We provide an overview with details about the methods in Tab. 11.

Overview of evaluated methods at 1024px. We provide an overview with details about the methods in Tab. 12.

6.5.2. Additional Reconstruction Results

In this section, we provide additional qualitative and quantitative results for the reconstruction experiments at 512 px and 1024 px. Unless otherwise stated, all results use the SWITTI backbone with our default settings.

512 px Reconstructions (COCO). We first compare reconstructions at 512 px resolution for three variants:
- SWITTI with quantization refinement (QR),
- SWITTI without QR, and
- TurboEdit [6].

Reconstructions are shown in fig. 12.

1024 px PSNR Comparison (OpenImages). At 1024 px, we report a method-level comparison in terms of PSNR over the OpenImages subset, including SWITTI w/ and w/o QR and all diffusion/flow baselines used in the main paper (TurboEdit, ReNoise, PnP, Ledits++, RF-Inversion, EditFriendly). The quantitative results can be seen in fig. 13.

1024 px Qualitative Comparison of QR. Finally, we provide a qualitative comparison at 1024 px between:
1. TurboEdit,
2. SWITTI without QR, and
3. SWITTI with QR.

This visualization highlights how QR specifically reduces blocky artifacts and restores sharpness in high-frequency regions without introducing over-smoothing (see fig. 14).

6.6. Details and qualitative samples of editing experiments

In our PIE-Bench editing experiments, we evaluate our method against recent diffusion-based and flow-based baselines. All baseline results reported in this paper were reproduced using the official code released by the respective authors, executed with the recommended default hyperparameters documented in their repositories.

Table 11. **Methods used in the 512px reconstruction experiments.** For each method we list the backbone model, the reconstruction procedure, and the exact settings used in our evaluation.

Method	Backbone	Procedure	Settings
Default SWITTI (baseline)	SWITTI [43]	VAR (no refinement)	s=6 (512px), CFG enabled for scales 2–8, default SWITTI sampling
TurboEdit [6]	SDXL [29]	Diffusion	4 denoising steps
ReNoise [11]	SD2.1 [32]	Diffusion	50 inversion + 50 inference steps
PnP [21]	SD1.4 [32]	Diffusion	50 inference steps
Ledits++ [2]	SD1.5 [32]	Diffusion	50 inversion steps
RF-Inversion [33]	FLUX-1 dev [23]	Rectified Flow	28 inversion steps
EditFriendly [19]	SD1.5 [32]	Diffusion	100 inversion steps

Table 12. **Methods used in the 1024px reconstruction experiments.** For each method we list the backbone model, the reconstruction procedure, and the exact settings used in our evaluation.

Method	Backbone	Procedure	Settings
Default SWITTI (baseline)	SWITTI [43]	VAR (no refinement)	s=8 (1024px), CFG enabled for scales 2–12, default SWITTI sampling
TurboEdit [6]	SDXL [29]	Diffusion	4 denoising steps
ReNoise [11]	SDXL [29]	Diffusion	50 inversion + 50 inference steps
PnP [21]	SD1.4 [32]	Diffusion	50 inference steps
Ledits++ [2]	SDXL [29]	Diffusion	50 inversion steps
RF-Inversion [33]	FLUX-1 dev [23]	Rectified Flow	28 inversion steps

Whenever multiple configuration presets or parameter options were provided, we evaluated the available variants and report the best-performing setting for each method for fair comparison. Tab. 13 contrains all the method configurations for the comparison of Tab. 1 in the main paper[2].

Tab. 14 lists all method configurations from Tab. 2.

Mask deactivation during style edits As discussed in Sec. 4, for style–transfer scenarios in the PIE benchmark we disable the masking mechanism entirely, both at 512px and 1024px resolution. Concretely, for all samples belonging to the category '9_change_style', we enforce full editing on the entire image by manually setting the editing mask to one, i.e., $\mathbf{M}_k = \mathbf{1}$ for all scales k. This ensures that stylistic transformations are applied globally, which is necessary because style edits typically require modifications across the entire image rather than localized changes.

[2]For discrete autoregressive approaches, official codebases were not publicly available. However, we reproduced the reported results for VARIN [5] and AREdit [45] following the descriptions in their papers. Performance metrics closely match the reported values, while efficiency numbers are recomputed based on our hardware setup (NVIDIA A6000).

Additional samples at 1024px. Fig. 15 shows additional editing results using the presented MLN approach.

6.7. Upscaled PIE-benchmark

Upsampling Strategy. To evaluate edits at 1024px resolution, we require high-quality high-resolution inputs. Since PIE is defined at 512px, we compare two upscaling strategies:

- simple linear interpolation
- diffusion-based super-resolution(in the main paper).

The two approaches yield different editing outcomes. Linear interpolation produces overly smooth textures and blurred edges, which propagate into the edited images and lead to less details and can introduce artifacts. In contrast, diffusion-based upsampling reconstructs sharper contours and plausible high-frequency structure, resulting in substantially more faithful and visually coherent edits.

Figure 12. **Reconstructions on COCO-512.** From left to right: input image, SWITTI w/o QR, TurboEdit, SWITTI w QR. QR reduces quantization artifacts and preserves local details compared to the baseline and diffusion-based reconstruction.

Table 13. Overview of evaluated editing methods, their backbone models, and the specific method configurations used in our experiments.

Method	**Backbone**	**Method Settings**
DDIM [36]	SD1.4 [32]	DDIM inversion with Prompt-to-Prompt cross-attention control.
ReNoise [11]	SDXL [29]	50 inversion steps + 50 inference steps.
LEdits++ [2]	SD1.5 [29]	50-step diffusion inversion.
EditFriendly [19]	SD1.5 [32]	DDPM inversion with 100 denoising steps.
PnP [21]	SD1.4 [32]	50-step diffusion inversion.
TurboEdit [6]	SDXL-Turbo [29]	4 denoising steps using SDXL-Turbo.

For all our 1024px experiments, we upsample the PIE images using InvSR [47] with 4 inference steps, and additionally provide the original PIE source prompt as conditioning to the diffusion-based upsampler.

Linear Interpolation Baseline. For completeness, we also evaluate all editing methods on a naive 1024px variant of the PIE benchmark obtained by linearly upsampling the original 512px images.

As shown in Tab. 15, linearly interpolated inputs in general lead to degraded background preservation and weaker text–image alignment compared to their diffusion-upsampled counterparts. These results further highlight that realistic high-frequency reconstruction—as provided by InvSR—is essential for fair and meaningful evaluation of editing performance at 1024px.

6.8. Recaptioning of OpenImages

The OpenImages subset used in our reconstruction evaluations (4.2) contains no textual annotations. To make it us-

Table 14. Overview of evaluated editing methods, their backbone models, and the specific method configurations used in our 1024px PIE experiments.

Method	Backbone	Method Settings
PnP [21]	SD1.4 [32]	50-step diffusion inversion.
LEdits++ [2]	SDXL [29]	50-step diffusion inversion.
ReNoise [11]	SDXL [29]	50 inversion + 50 inference steps.
TurboEdit [6]	SDXL [29]	4 denoising/editing steps.
RF-Inversion [33]	Flux 1.dev [23]	28-step rectified-flow inversion.

Table 15. **Quantitative evaluation (interpolated, i.e. without InvSR) on PIE-1024.** We report background preservation (PSNR, LPIPS), separate text–image alignment scores (CLIP similarity on whole image and edited region), and wall-clock runtime. Best values are **bold**.

Method	PSNR ↑	LPIPS ↓	CLIP Whole ↑	CLIP Edited ↑	Wall (s) ↓
PnP [21]	17.67	161.96	24.62	23.20	17
LEDITS++ [2]	22.32	58.65	24.17	22.06	18.4
ReNoise [11]	22.33	87.33	23.32	21.11	13.2
TurboEdit [6]	**28.13**	43.53	25.13	22.56	4.1
RF-Inversion [33]	20.62	147.41	24.41	25.11	22.3
Ours	27.98	**37.76**	**25.84**	**23.15**	**1.6**

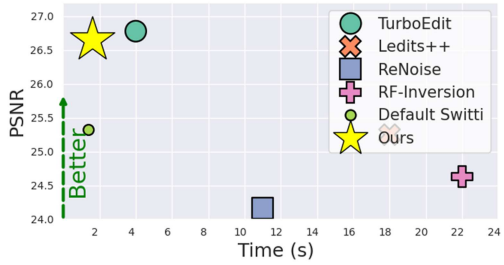

Figure 13. **PSNR of different reconstruction methods at 1024 px.** SWITTI with QR achieves the highest PSNR, outperforming both the non-refined SWITTI baseline and diffusion/flow-based methods.

able for text-conditioned training, we automatically generate for each image a language caption using GPT-4V [49]. In Fig. 16 we show 3 recaptioned sample images.

6.9. Additional qualitative editing samples

Additional qualitative editing results at 512px and 1024px can be seen in Fig. 17 and Fig. 15 respectively.

6.10. More ablations

Applicability to other VAR backbones To evaluate the generality of *Masked Logit Nudging*, we apply it to the Infinity model [12] without any retraining or architectural modification. All experiments are conducted at 512×512 resolution with the 2B parameter checkpoint. Our main goal is not to optimize Infinity but to verify that MLN transfers across VAR backbones.

Therefore we keep the default infinity hyperparameters identical to those used in the official code and our Switti+MLN implementation. The main difference is that we change $k_{cut} = 4$ in the nudging schedule applied.

Due to the difference in quantization schemes—Euclidean residual quantization in SWITTI versus binary spherical quantization (BSQ) in Infinity—we cannot apply the reconstruction enhancement described in Sec. 3.4. Thus, Infinity relies solely on the MLN editing mechanism.

We provide representative examples for Ledits++ [2] and Infinity+MLN (see fig. 18), demonstrating that MLN reliably transfers across architectures despite quantizer differences.

Precision–Efficiency Trade-Off We analyze the impact of numerical precision on runtime and reconstruction fidelity. As shown in Tab. 16, switching from `float32` to `float16` substantially accelerates inference—down to only ~ 0.28 s per edit—while maintaining competitive reconstruction quality.

Importantly, **SWIFTEdit** [28], the current state of the art in *fast* image editing, achieves comparable speed but *requires additional training*, whereas our method is entirely **training-free** and still delivers noticeably better reconstruction fidelity at nearly identical runtime.

Finally, we also evaluate `float16` at **1024px** resolution and observe that performance remains stable, confirming that half-precision maintains editability even at high resolutions.

Figure 14. **Qualitative reconstruction comparison at 1024 px.** From left to right: input image, TurboEdit, SWITTI w/o QR, SWITTI w/ QR. Quantization refinement yields visibly sharper reconstructions and fewer artifacts, especially in textures and edges.

Table 16. **Precision–efficiency ablation on the PIE-Benchmark.** Comparison of editing speed and reconstruction quality for `float16` and `float32`. Our method outperforms SWIFTEdit [28] significantly in overall editing performance, while only requiring 40ms longer and being training-free.

Method	Time/Edit (s)	PSNR ↑	MSE ↓	CLIP ↑
Ours – float16 (512px)	0.28	28.01	41.23	22.01
Ours – float32 (512px)	0.82	29.70	23.30	22.72
SWIFTEdit [28] – trained, float16	0.23	23.31	61.80	21.91
Ours – float16 (1024px)	0.46	24.80	74.10	22.95
Ours – float32 (1024px)	1.60	26.70	45.51	23.67

6.11. Failure cases

While our method achieves strong editing consistency across most scenarios, we observe that the majority of failure cases arise from mask inaccuracies. Since Masked Logit Nudging (MLN) and the Quantization refinement relies on spatial guidance to determine where logits should be nudged toward the source or target distribution, misaligned masks can propagate directly into visible artifacts.

Incorrect fine-grained masks. In several challenging examples, the cross-attention–based mask incorrectly assigns high-confidence editing regions to pixels that should remain untouched. Figure 19 illustrates such a case: when editing a cat into a bear, the mask partially overlaps with the cat's whiskers and nose hair. As a result, the model unintentionally replaces thin facial details with textures from the target concept, leading to unnatural blending.

Structural errors from coarse masks. A second class of failures emerges when the mask captures the correct semantic region but is spatially too coarse. In the couch-editing example, the model attempts to preserve the original geometry, but the spatial mask extends into the background and occludes a portion of the sofa boundary. Consequently, the autoregressive refinement reconstructs a distorted or incomplete couch—either flattening the cushion or introducing inconsistent shading at the edges. These errors confirm that token-level masking must be both semantically accurate and spatially sharp to avoid disrupting the low-frequency structure encoded in the early scales.

Discussion. Across both categories, we find that mask quality remains the dominant factor limiting worst-case performance. Since MLN itself operates correctly when-

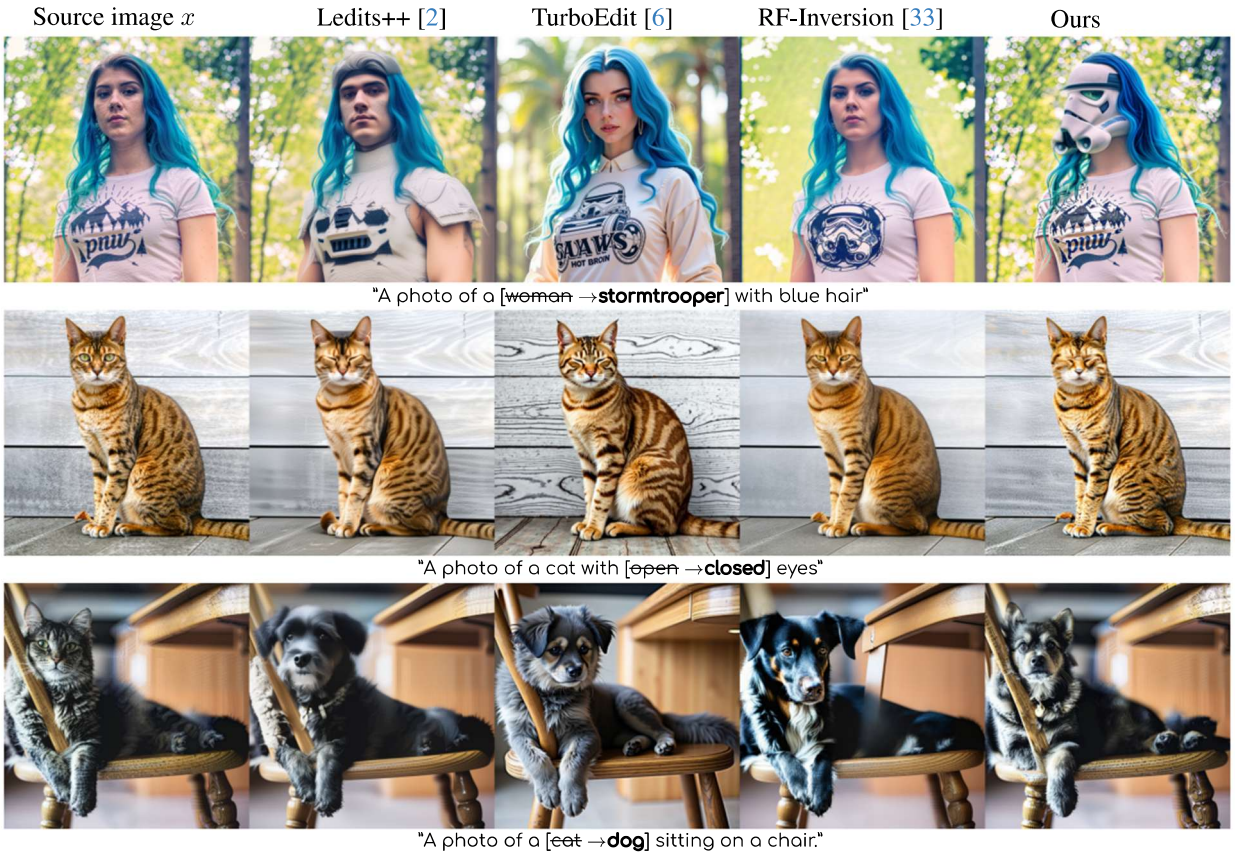

Figure 15. **Qualitative editing results on PIE-1024.** Comparison of Masked Logit-nudging(Ours), Ledits++ [2], TurboEdit[6] and RF-Inversion [33].

"A classic light blue sedan parked on a grassy field."

"An aerial view of a large office building and trees along the street."

"A close-up of a caramel flan dessert on a white plate."

Figure 16. **Examples from OpenImages with automatically generated recaptions.**

ever the preserved region is well specified, improving the mask—e.g., by integrating multi-scale attention cues or leveraging segmentation priors—is likely to further reduce these failure modes without modifying the underlying nudging mechanism.

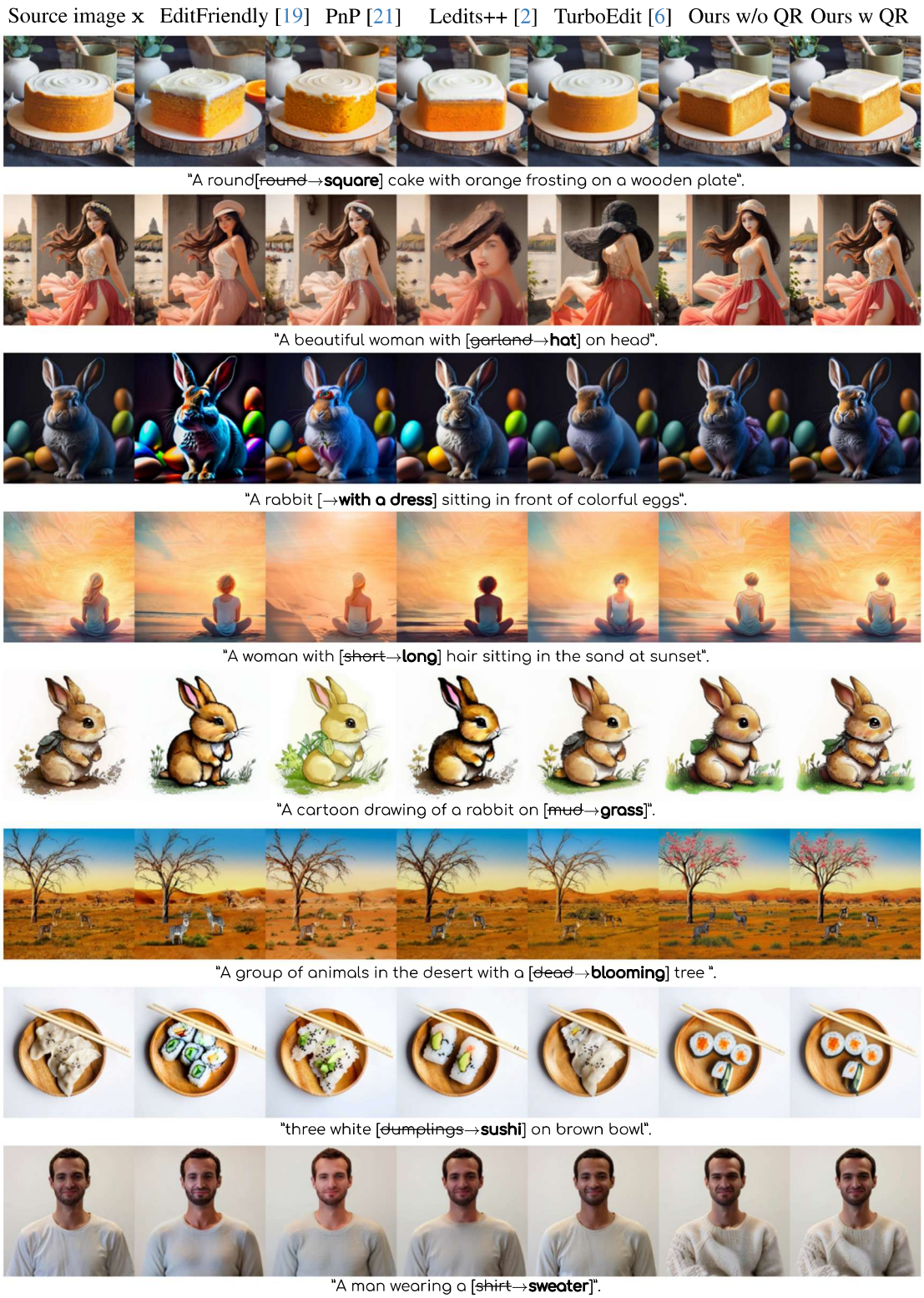

Figure 17. **Additional qualitative results on PIE-512.** Editing results of EditFriendly [19], PnP [21], Ledits++ [2], TurboEdit [6], and our proposed Masked Logit Nudging without Quantization refinement (Ours w/o QR) and with Quantization refinement (Ours w QR).

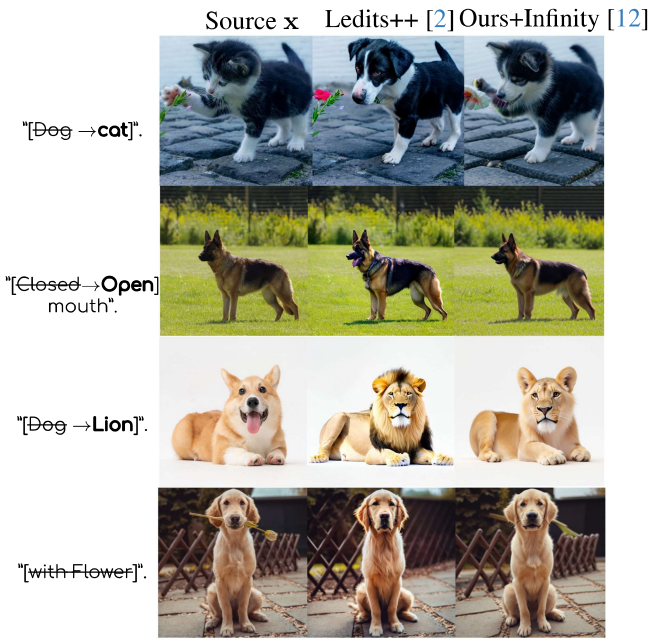

Figure 18. **Qualitative results on Infinity.** Our approach translates seemlessly to other VAR backbones such as Infinity [12].

(a) Whisker-level masking error. When editing a cat into a bear, the mask spills over into the nose hair and whisker regions. As MLN nudges logits inside these pixels, the model unintentionally replaces thin facial details with bear-like textures, producing unnatural local artifacts.

(b) Coarse mask affecting structure. In this example, the spatial mask extends beyond the edited object and partially covers the sofa boundary. As a result, the AR refinement fails to reconstruct the correct geometry, leading to a warped cushion and inconsistent shading along the couch silhouette.

Figure 19. **Masking-related failure cases.** Most of our failure modes originate from inaccurate or overly coarse masks. Because MLN modifies logits only inside the predicted editing region, even slight mask misalignments introduce noticeable artifacts—especially in high-frequency areas such as whiskers, fur, or object boundaries. Improving mask precision directly reduces these errors without modifying the underlying editing mechanism.