Delta-Adapter: Scalable Exemplar-Based Image Editing with Single-Pair Supervision

Jiacheng Chen[1] **Songze Li**[1] **Han Fu**[1] **Baoquan Zhao**[1] **Wei Liu**[3]
Yanyan Liang[4] **Qing Li**[5] **Xudong Mao**[1,2*]
[1]Sun Yat-sen University [2]Shanghai University of Finance and Economics [3]Video Rebirth
[4]Macau University of Science and Technology [5]The Hong Kong Polytechnic University

Abstract

Exemplar-based image editing applies a transformation defined by a source-target image pair to a new query image. Existing methods rely on a pair-of-pairs supervision paradigm, requiring two image pairs sharing the same edit semantics to learn the target transformation. This constraint makes training data difficult to curate at scale and limits generalization across diverse edit types. We propose Delta-Adapter, a method that learns transferable editing semantics under single-pair supervision, requiring no textual guidance. Rather than directly exposing the exemplar pair to the model, we leverage a pre-trained vision encoder to extract a semantic delta that encodes the visual transformation between the two images. This semantic delta is injected into a pre-trained image editing model via a Perceiver-based adapter. Since the target image is never directly visible to the model, it can serve as the prediction target, enabling single-pair supervision without requiring additional exemplar pairs. This formulation allows us to leverage existing large-scale editing datasets for training. To further promote faithful transformation transfer, we introduce a semantic delta consistency loss that aligns the semantic change of the generated output with the ground-truth semantic delta extracted from the exemplar pair. Extensive experiments demonstrate that Delta-Adapter consistently improves both editing accuracy and content consistency over four strong baselines on seen editing tasks, while also generalizing more effectively to unseen editing tasks. Code will be available at `https://delta-adapter.github.io`.

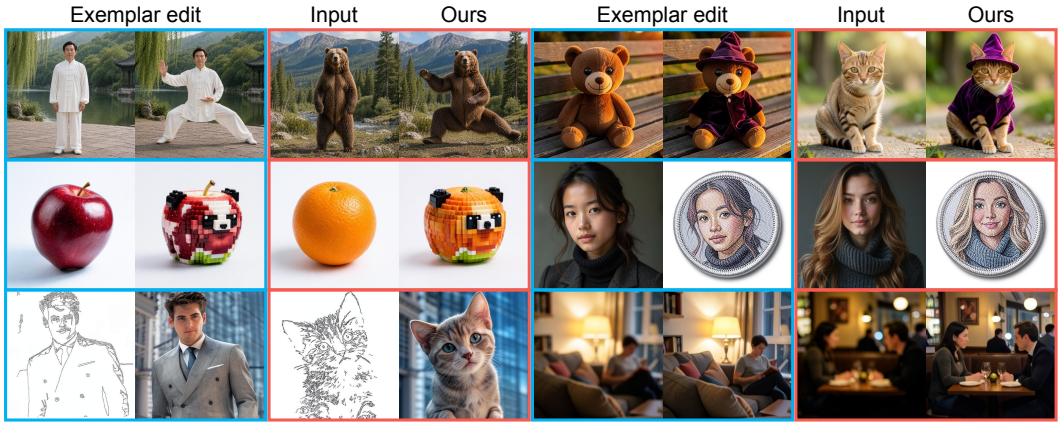


Figure 1: Exemplar-based image editing with Delta-Adapter. Our method learns complex transformations from exemplar image pairs and faithfully applies them to new input images.

*Corresponding author (xudong.xdmao@gmail.com).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Instruction-based image editing [7, 28] has demonstrated powerful and flexible image manipulation through natural language. However, certain edits, such as subtle appearance shifts and edit extent, are inherently difficult to articulate precisely in words. This limitation motivates exemplar-based image editing [5, 49], also known as image analogy [19], where a source/target exemplar pair defines the desired transformation, which is then applied analogously to a new query image. Compared to text instructions, exemplar pairs convey editing intent more directly and unambiguously.

Existing exemplar-based editing methods [49, 16, 30] predominantly adopt a *pair-of-pairs supervision* paradigm: given two image pairs $\{a, a'\}$ and $\{b, b'\}$ sharing the same edit semantics, the model learns to predict b' from the tuple (a, a', b) by transferring the transformation observed in $\{a, a'\}$. Despite its effectiveness, this formulation is inherently restrictive. To reliably isolate the intended edit, both pairs must exhibit closely matched transformations, and any uncontrolled discrepancy can introduce ambiguity that undermines learning. This strict alignment requirement makes training data difficult to curate and scale, limiting the diversity of learnable edit types and the model's generalization capacity. Moreover, existing methods often rely on textual guidance at both training and inference time, making performance sensitive to prompt wording and imposing an extra burden on users.

These limitations raise a central question: Can transferable editing semantics be learned under *single-pair supervision*, without textual guidance? The reliance on two pairs in existing methods stems from a specific architectural choice: the model is conditioned on the complete exemplar pair $\{a, a'\}$, directly exposing the edited image a' as input. Because the target appearance is fully observable, a second pair becomes necessary to supervise the prediction of b'. Our key insight is to adopt a fundamentally different conditioning strategy. Rather than exposing a' directly, we extract a semantic delta $\Delta_{a\to a'}$ that encodes the visual transformation from a to a', and condition the model solely on the tuple $(a, \Delta_{a\to a'})$. Since a' is never directly visible to the model, it can serve as the prediction target, enabling single-pair supervision without requiring additional exemplar pairs.

We instantiate this idea in Delta-Adapter, a framework for exemplar-based image editing under single-pair supervision that requires no textual guidance. Given a single exemplar pair $\{a, a'\}$, we leverage a pre-trained vision encoder [59] to compute a semantic delta $\Delta_{a\to a'}$ that encodes the visual transformation between the two images. This delta is injected into a pre-trained image editing model via a Perceiver-based adapter [1]. During training, only the adapter parameters are optimized to reconstruct a' from the tuple $(a, \Delta_{a\to a'})$, while the base editing model remains entirely frozen. To further improve editing fidelity, we introduce a semantic delta consistency loss that encourages the feature-space displacement between the source and generated images to align with the ground-truth semantic delta.

Our proposed single-pair supervision paradigm offers two key practical advantages. First, training requires only individual source/target image pairs, enabling direct use of existing large-scale image editing datasets. This substantially broadens the diversity of edit types seen during training and improves generalization to unseen edits. Second, the single-pair paradigm naturally enables a test-time adaptation strategy: for challenging unseen exemplars, Delta-Adapter can be efficiently fine-tuned on the provided image pair to better capture the intended transformation.

We validate Delta-Adapter through extensive qualitative and quantitative experiments, comparing against four strong baselines across a diverse range of editing tasks. On seen editing tasks, our method achieves superior editing accuracy while better maintaining content consistency. Moreover, Delta-Adapter exhibits better generalization to unseen edits compared to all baselines. When further equipped with the test-time adaptation strategy, performance on unseen tasks improves substantially, reaching levels comparable to those achieved on seen tasks.

2 Related Work

Diffusion-based image editing. Diffusion models have emerged as the dominant paradigm for high-quality image generation and editing [20, 45, 27, 42], and a rich body of work has explored how diverse conditioning signals can guide the editing process. Text-conditioned methods are among the most widely adopted, conveying desired changes through natural language [35, 18, 25, 50, 40, 41, 6, 7, 15, 46, 62, 14, 21, 24, 9]. While language affords flexible semantic control, it often struggles to precisely capture subtle appearance changes, fine-grained spatial extents, or complex transformations.

Mask-conditioned methods address the localization challenge by restricting edits to user-specified regions [2, 3, 12, 58, 52, 65], while structure-conditioned methods further enforce spatial faithfulness by incorporating geometric cues such as edges, depth, or pose [60, 36, 64]. Reference-guided methods take a complementary approach, transferring appearance, identity, or style from a reference image to the target [56, 10, 29, 57]. Exemplar-based image editing methods [5, 37, 49] condition the model on a before-and-after image pair that jointly defines the desired transformation.

Exemplar-based image editing. The idea of learning visual transformations from image pairs traces back to the classical image analogy framework [19], and has regained significant attention with the rise of large generative models [5, 63, 49, 30]. Existing diffusion-based approaches can be broadly categorized by how they leverage the exemplar pair at test time. Optimization-based methods adapt learnable parameters to encode the transformation defined by the pair [37, 23, 33]. While capable of capturing fine-grained edits, these methods require a costly per-edit optimization process. Training-free methods avoid this overhead by exploiting the in-context reasoning capabilities of pre-trained diffusion models [17, 48]. Training-based methods instead learn a general editing policy from data, enabling efficient inference without test-time optimization [49, 53, 47, 54, 30, 16, 34]. However, existing training-based approaches rely on pair-of-pairs supervision: two image pairs sharing the same edit semantics are required, where the model observes the transformation in one pair and is trained to predict the target image in the other. This requirement makes training data difficult to curate and scale. Our method addresses this limitation by conditioning on the semantic delta rather than the full exemplar pair, enabling single-pair supervision. Although ReEdit [48] also extracts a semantic delta for exemplar-based editing, the two methods differ in fundamental ways. First, ReEdit operates as a training-free method, whereas ours is a trained model. Second, ReEdit conditions on a combination of the semantic delta and the target image representation, while our method conditions solely on the delta, explicitly decoupling the edit operation from image content. Third, ReEdit projects the semantic delta into the textual embedding space and fuses it with a text prompt for conditioning, whereas our model injects the delta directly into the editing backbone via a Perceiver-based adapter.

3 Preliminary

Rectified flow. Our model builds upon FLUX, which formulates image generation as a rectified flow [31, 32] process in the latent space. Let z_0 denote a clean image latent and $z_1 \sim \mathcal{N}(0, I)$ a noise latent. Rectified flow defines a straight-line interpolation between z_0 and z_1 as $z_t = (1-t)\, z_0 + t\, z_1$, where $t \in [0, 1]$. A velocity network v_θ is trained to predict the constant target velocity $z_1 - z_0$ along this trajectory, conditioned on the noisy latent z_t, the timestep t, and a text prompt c:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{t, z_0, z_1, c}\left[\left\| v_\theta(z_t, t, c) - (z_1 - z_0) \right\|_2^2\right]. \tag{1}$$

During training, a coarse estimate of the clean latent can be recovered from the predicted velocity as

$$\hat{z}_0 = z_1 - v_\theta(z_t, t, c). \tag{2}$$

4 Method

4.1 Problem Formulation

We address the task of exemplar-based image editing under single-pair supervision. Given a single exemplar pair $\{a, a'\}$, where a is the source image and a' is its edited counterpart, our goal is to learn the visual transformation $a \to a'$ and apply it analogously to an unseen query image b, producing an edited output $\hat{b}'$, without any textual guidance at training or inference time.

The key distinction between our formulation and prior work lies in the conditioning input exposed to the model. Existing methods [16, 34] condition the editing model on the full exemplar pair $\{a, a'\}$, making the target image a' directly observable. In this setting, supervising the model on the same pair is ill-posed: the desired edited appearance is already present as a conditioning input. Prior methods therefore rely on a second aligned pair $\{b, b'\}$ to supervise whether the edit inferred from $\{a, a'\}$ transfers to b. This pair-of-pairs requirement makes training data difficult to curate and fundamentally limits the scalability of model training.

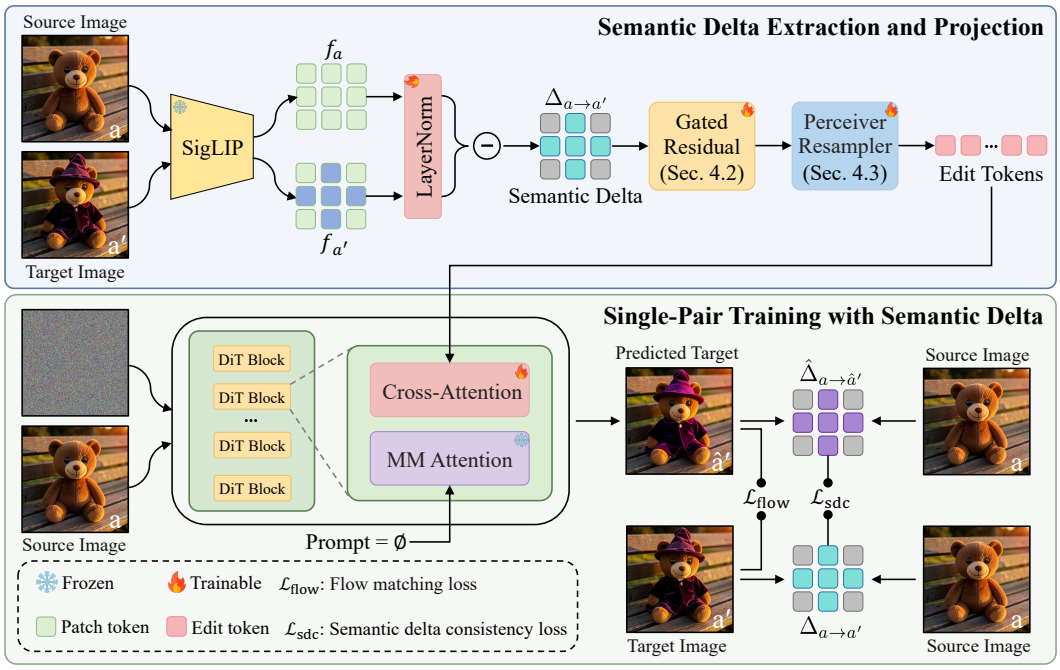


Figure 2: **Overview of Delta-Adapter.** Given a single exemplar pair $\{a, a'\}$, we first extract patch-level SigLIP features and compute a normalized semantic delta $\Delta_{a\to a'} = \mathrm{LN}(f_{a'}) - \mathrm{LN}(f_a)$. The delta is refined via a gated residual projection and converted into edit tokens through a Perceiver resampler. These tokens are injected into a frozen DiT-based editing backbone via decoupled cross-attention to reconstruct the target image. Training is supervised by a flow-matching loss combined with a semantic delta consistency loss, which encourages the predicted edit to align with the ground-truth semantic transformation.

Our key insight is to condition the model on an explicit *semantic delta* $\Delta_{a\to a'}$ that encodes the transformation from a to a', rather than on a' itself. Formally, the model takes the tuple $(a,\ \Delta_{a\to a'})$ as input and is trained to reconstruct a':

$$\hat{a}' = \mathcal{F}_\theta(a,\ \Delta_{a\to a'})\,. \tag{3}$$

Unlike prior methods that expose the full edited image a' as a condition, our model receives only a semantic displacement $\Delta_{a\to a'}$. This prevents direct copying of the target appearance while retaining the transformation signal necessary for supervision. Consequently, each single image pair can supervise itself, without requiring an additional aligned pair.

As illustrated in Figure 2, our framework operates as follows. Given the exemplar pair $\{a, a'\}$, we first extract a normalized semantic delta $\tilde{\Delta}_{a\to a'}$ (Section 4.2). This delta is then resampled into a sequence of edit tokens and injected into the pre-trained editing backbone (Section 4.3). The model is trained to reconstruct a' using a flow matching loss augmented by a semantic delta consistency loss that enforces alignment between the predicted and ground-truth edit directions (Section 4.4).

4.2 Semantic Delta Extraction

The first step is to construct a representation of the visual transformation $a \to a'$. We describe this in two stages: computing a normalized token-level semantic delta, and refining it via a gated residual projection.

Normalized semantic delta. Given the exemplar pair $\{a, a'\}$, we employ a pre-trained SigLIP [59] encoder to extract dense patch-level features $f_a,\ f_{a'} \in \mathbb{R}^{L\times D_r}$, where L is the number of patch tokens and D_r their dimensionality. Specifically, we extract the last hidden states before the pooling layer, preserving the per-patch spatial structure that is essential for image editing. A natural first attempt is to define the semantic delta as the naive difference $f_{a'} - f_a$. However, this formulation is often dominated by instance-dependent magnitude variations in the raw SigLIP feature space. To

address this, we apply token-wise layer normalization [4] before differencing:

$$\Delta_{a\to a'} = \mathrm{LN}(f_{a'}) - \mathrm{LN}(f_a), \tag{4}$$

where $\mathrm{LN}(\cdot)$ normalizes each token independently. This suppresses instance-level magnitude variation while preserving the directional change in feature space.

Gated residual refinement. Even after normalization, $\Delta_{a\to a'}$ may still contain task-irrelevant variation or imprecisely aligned edit directions. We therefore introduce a gated residual projection to further refine the edit signal:

$$\tilde{\Delta}_{a\to a'} = \Delta_{a\to a'} + \tanh(g)\,\mathrm{Linear}(\Delta_{a\to a'}), \tag{5}$$

where $\mathrm{Linear}(\cdot)$ is a token-wise affine transformation shared across all patch tokens, and $\tanh(g)$ is a bounded learnable scalar gate. The gate is initialized to zero, ensuring the model first learns a stable semantic delta representation before gradually incorporating residual corrections.

Discriminability of semantic deltas. To evaluate whether semantic deltas capture information that distinguishes editing tasks, we conduct a linear probing analysis covering all 218 editing tasks in the Relation dataset [16]. Specifically, we extract semantic deltas from distinct pairs of source and edited images and train a linear classifier to predict the corresponding editing task. The classifier achieves a top-1 accuracy of 98.01%, demonstrating that semantic deltas encode highly discriminative information across diverse editing transformations.

4.3 Semantic Delta Projection and Injection

Given the extracted semantic delta $\tilde{\Delta}_{a\to a'}$, we project it into a fixed-length sequence of conditioning tokens and inject them into the DiT-based editing backbone.

Perceiver-based resampling. Prior IP-Adapter-style methods [57, 16] map visual encoder features into the generative model via global average pooling followed by an MLP. For exemplar-based editing, however, we find this design generalizes poorly to unseen tasks. We attribute this limitation to the pooling operation: collapsing $\tilde{\Delta}_{a\to a'}$ into a single global vector discards the localized and relational changes that are critical for faithfully representing the intended edit. To address this, we replace the pooling-MLP with a Perceiver resampler [1]. Specifically, N learnable query tokens cross-attend to the full patch sequence of $\tilde{\Delta}_{a\to a'}$, producing a fixed-length edit representation $R \in \mathbb{R}^{N\times D_r}$. Unlike global average pooling, which treats all patches uniformly, the cross-attention mechanism can exploit the positional information inherent in SigLIP patch tokens when aggregating edit signals.

Per-token projection. A common practice for mapping R into the conditioning space of the DiT blocks [42] is to use a shared linear projection for all tokens [1]. We find this shared mapping overly restrictive for exemplar-based editing: because each token in R is expected to encode a distinct aspect of the edit, a uniform projection suppresses such specialization. We therefore assign each latent token its own affine projection, $e_i = W_i\, r_i + b_i$ for $i = 1, \ldots, N$, where $r_i \in \mathbb{R}^{D_r}$ is the i-th token of R and (W_i, b_i) are token-specific learnable parameters. The resulting edit tokens $\{e_i\}_{i=1}^{N}$, stacked into $E \in \mathbb{R}^{N\times D_c}$, form the final conditioning representation passed to the editing backbone.

Our Perceiver resampler with per-token projection offers two key advantages. First, as demonstrated in Table 2, it improves both editing accuracy and content preservation, with particularly pronounced gains on unseen tasks. Second, it is more parameter-efficient, requiring only half the parameters of the pooling-MLP projection employed in [16].

Decoupled attention injection. Following [57, 16], we inject the edit tokens E into each DiT block via a decoupled cross-attention branch. Specifically, we introduce learnable key and value projections $K_\Delta = EW_k^\Delta$ and $V_\Delta = EW_v^\Delta$, and compute the branch output as $Z_\Delta = \mathrm{Softmax}\Big(QK_\Delta^\top/\sqrt{d}\Big)\,V_\Delta$, where Q denotes the query from the original DiT branch. The branch output is then fused with the original attention output via a residual connection: $Z_{\mathrm{new}} = Z + \lambda_{\mathrm{ca}}\, Z_\Delta$, where λ_{ca} is a scalar controlling the injection strength, fixed to 1.0 during training. During training, only W_k^Δ, W_v^Δ, and the preceding projection layers are optimized, while all backbone weights remain frozen.

4.4 Semantic Delta Consistency Loss

Our training objective consists of two loss terms. The first applies the flow matching loss (Eq. 1) to reconstruct the target image a'. The second is an auxiliary semantic delta consistency loss that provides explicit supervision over the edit semantics.

At each training step, we estimate the denoised latent $\hat{z}_0$ via Eq. 2 and decode it through the VAE decoder to obtain the reconstructed image $\hat{a}'$ in pixel space. We then extract patch-level features $f_{\hat{a}'}$ from $\hat{a}'$ using the SigLIP encoder, and compute the predicted semantic delta as $\hat{\Delta}_{a\to\hat{a}'} = \mathrm{LN}(f_{\hat{a}'}) - \mathrm{LN}(f_a)$. Notably, since our backbone model performs denoising in only four steps, the recovered $\hat{z}_0$ is sufficiently sharp to support reliable feature extraction even at the very first denoising step, as illustrated in Figure 23.

Since an edit often affects only a subset of image regions, patches undergoing large semantic shifts should exert a stronger supervisory signal than those that remain nearly unchanged. We therefore assign each patch token ℓ a weight $m^{(\ell)} = \|\Delta^{(\ell)}_{a\to a'}\|_2 \,/\, \max_j \|\Delta^{(j)}_{a\to a'}\|_2$ proportional to its relative magnitude of change in the ground-truth delta $\Delta_{a\to a'}$. The semantic delta consistency loss then minimizes the patch-weighted cosine distance between the predicted and ground-truth deltas:

$$\mathcal{L}_{\mathrm{sdc}} = 1 - \frac{1}{L}\sum_{\ell=1}^{L} m^{(\ell)} \cos\Big(\Delta^{(\ell)}_{a\to a'},\ \hat{\Delta}^{(\ell)}_{a\to\hat{a}'}\Big), \tag{6}$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity. This objective encourages the model to produce edits whose semantic deviation from the source image aligns directionally with the intended edit direction.

The full training objective combines both terms:

$$\mathcal{L} = \mathcal{L}_{\mathrm{flow}} + \lambda_{\mathrm{sdc}}\,\mathcal{L}_{\mathrm{sdc}}, \tag{7}$$

where λ_{sdc} controls the relative contribution of the semantic consistency term.

4.5 Test-Time Adaptation

Despite the generalization benefits of large-scale training under our single-pair supervision paradigm, the model may still struggle to capture fine-grained details on particularly challenging unseen tasks. A key advantage of our paradigm is that it naturally supports test-time adaptation using only the exemplar pair provided at inference. Concretely, we fine-tune Delta-Adapter for a small number of gradient steps (20 in our experiments) using the objective in Eq. 7. This stands in contrast to pair-of-pairs methods, which require an additional aligned pair for fine-tuning. As demonstrated in Section 5.2, test-time adaptation yields substantial improvements on challenging unseen tasks. To ensure fair comparison, test-time adaptation is not applied in any comparisons with baselines.

5 Experiments

5.1 Implementation and Evaluation Setup

Training data. Since Delta-Adapter requires only single-pair supervision, it can readily leverage existing training datasets designed for instruction-based image editing. Specifically, we train our model on approximately one million image pairs drawn from three sources: Relation [16], Pico-Banana [43], and NHR-Edit [26]. For the evaluation of seen tasks, we train exclusively on 16K image pairs from the Relation dataset to ensure a fair comparison with the baselines.

Implementation details. Our implementation builds upon the publicly available FLUX.2-klein-4B model, with SigLIP [59] serving as the image encoder. During training, both λ_{sdc} and λ_{ca} are fixed to 1.0. The model is trained for 100K steps on 4 $\times$ H200 GPUs with a per-GPU batch size of 16, using AdamW with a learning rate of 1×10^{-4} in bfloat16 precision. More implementation details for our method and all baselines are provided in Appendix A.

Baselines. We compare our method against four representative baselines: RelationAdapter [16], LoRWeB [34], VisualCloze [30], and Edit Transfer [8]. In Appendix F, we further include comparisons with two multimodal image generation models, Nano Banana 2 [13] and GPT-Image-2 [38], as well as an optimization-based method, PairEdit [33].

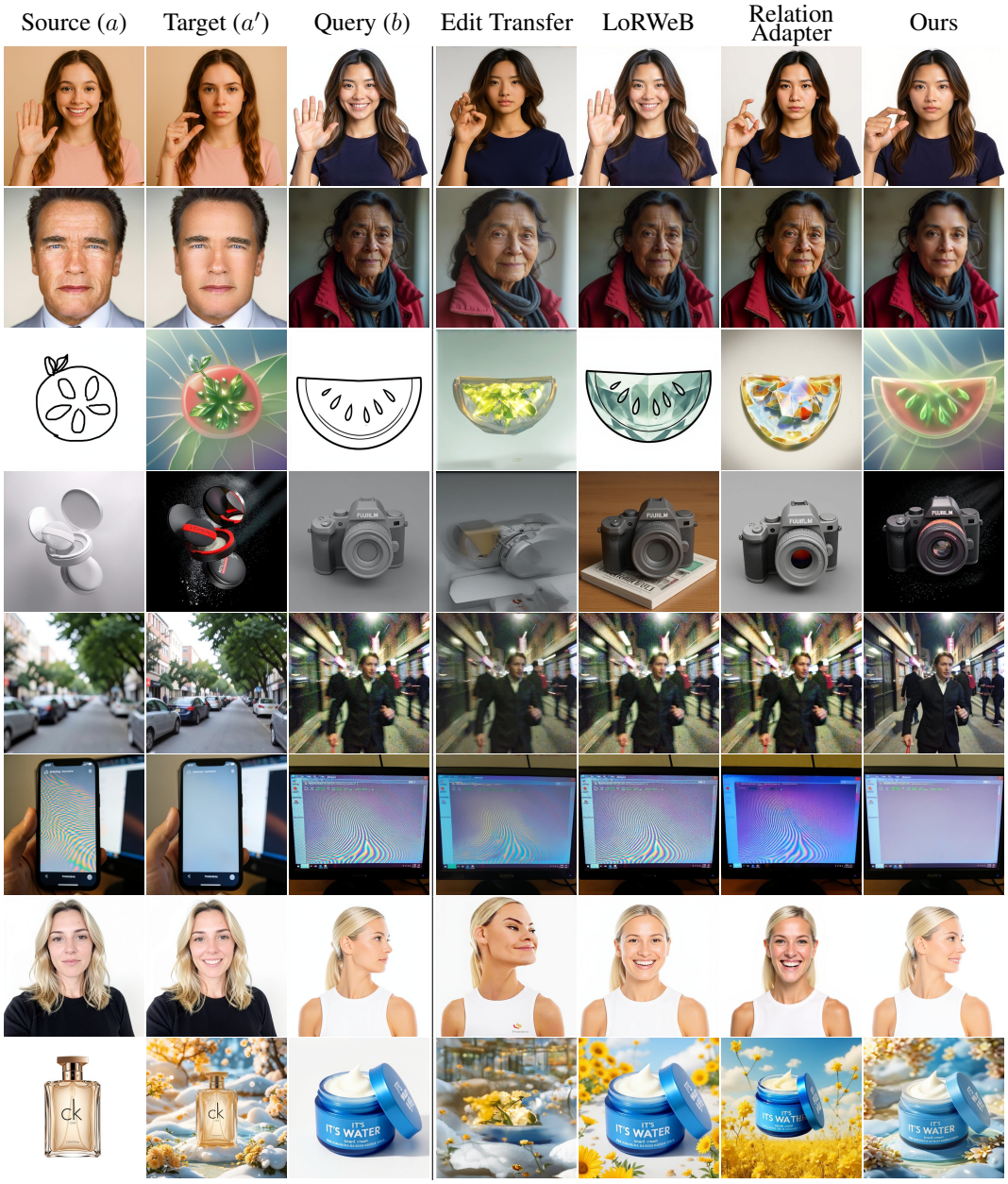


Figure 3: **Qualitative comparison on seen editing tasks.** We compare Delta-Adapter with Relation-Adapter [16], LoRWeB [34], and Edit Transfer [8]. Delta-Adapter more faithfully captures the edit semantics implied by the exemplar pair and applies them to the query image, while better preserving its underlying structure and identity.

Evaluation protocol. We adopt LPIPS [61] and CLIP-I [44] to measure perceptual similarity and semantic alignment between the edited images and either the source images, or the ground-truth images when available (e.g., those provided in the Relation dataset [16]). To evaluate editing accuracy, we employ the CLIP Direction Score (CDS) [37], which measures the alignment between the exemplar edit direction ($a \rightarrow a'$) and the predicted edit direction ($b \rightarrow \hat{b}'$) in CLIP feature space. In addition, we leverage GPT-5.4 to assess two aspects of each edited result on a 5-point scale: content consistency in unedited regions (GPT-C) and editing accuracy with respect to the exemplar pair (GPT-A). Appendix H provides further details on the GPT-based metrics, along with reliability analyses based on correlations with human ratings and robustness analyses using alternative evaluators and prompts. For seen tasks, we evaluate across all 218 tasks in the Relation dataset with

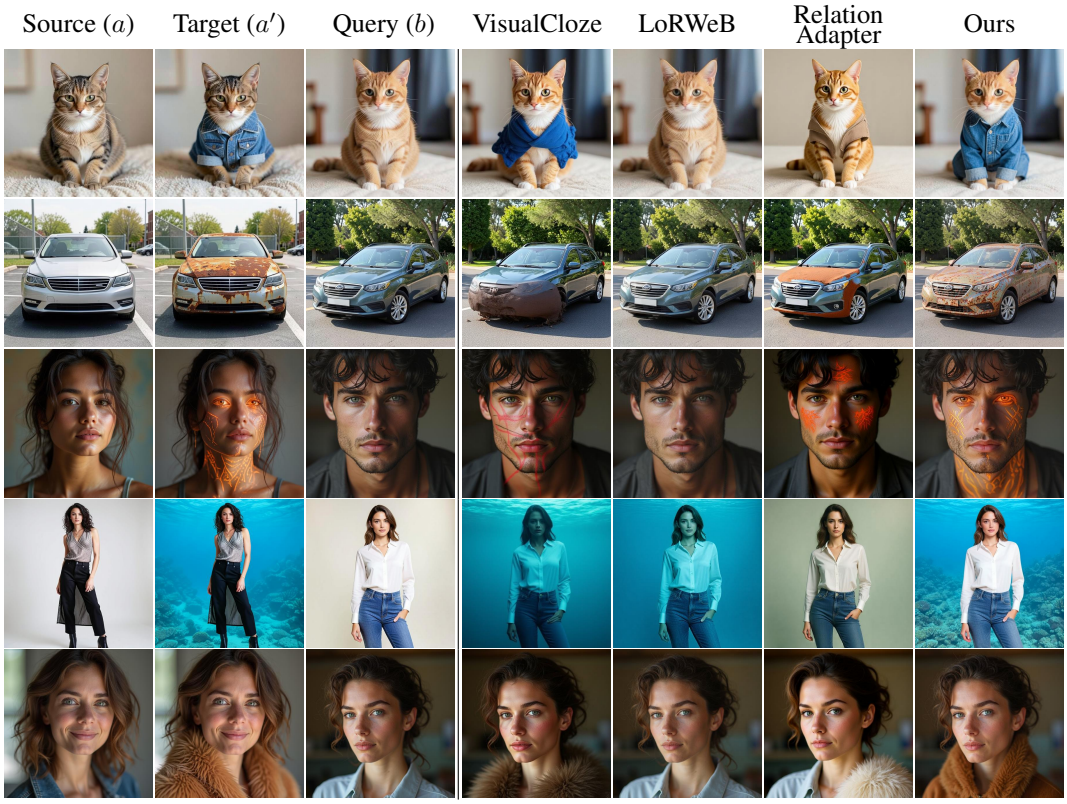


Figure 4: **Qualitative comparison on unseen editing tasks.** We compare Delta-Adapter with RelationAdapter [16], LoRWeB [34], and VisualCloze [30]. Across diverse unseen transformations, Delta-Adapter produces outputs that are semantically aligned with the exemplar edit.

5 query images each, yielding 1,090 generations per method. For unseen tasks, in addition to the unseen validation set from RelationAdapter, which consists of relatively simple tasks, we further construct 50 novel tasks spanning style transfer, attribute change, and object transformation, with 5 query images per task, yielding 250 generations per method.

5.2 Results

Qualitative evaluation. Figure 3 presents qualitative comparisons on seen tasks, where all models are trained on the Relation dataset [16]. LoRWeB [34] often fails to capture the intended transformation from the exemplar pair, producing outputs that are nearly identical to the query image (rows 1 and 2). Edit Transfer [8] similarly struggles to infer the edit semantics and frequently yields outputs with degraded visual quality. We further investigate the applicability of single-pair supervision to Edit Transfer and LoRWeB, with qualitative results provided in Appendix E. RelationAdapter [16] achieves improved editing fidelity but still falls short on challenging tasks such as motion deblurring (row 5) and image demoiréing (row 6), and struggles to maintain content consistency with the query image (rows 1 and 3). In contrast, Delta-Adapter reliably applies the inferred edit semantics to the query image while maintaining content consistency. We observe that Delta-Adapter successfully learns all editing tasks present in the Relation dataset. As further demonstrated in Figure 4, this advantage extends to unseen tasks, where Delta-Adapter exhibits stronger generalization than all baselines. Notably, all baselines require textual instructions at both training and inference time, whereas our method operates without any textual guidance yet achieves superior performance.

Beyond the editing tasks in the Relation dataset, we demonstrate results on more complex editing scenarios in Figures 1 and 16. Furthermore, Delta-Adapter supports continuous image editing by adjusting the injection strength λ_{ca} of the decoupled cross-attention, as shown in Figure 5. This capability arises from the fact that our method injects only the editing signal, whereas methods that inject the full exemplar pair (e.g., RelationAdapter) are unable to achieve such continuous control. Additional qualitative results are provided in Appendix C.

Table 1: **Quantitative comparison on seen and unseen editing tasks.** Ours-16K denotes our model trained on 16K image pairs from the Relation dataset, and Ours-1M is trained on 1M image pairs. VisualCloze is not reported on seen-task evaluation due to training instability on the Relation dataset.

Method	CLIP-I↑	LPIPS↓	CDS↑	GPT-C↑	GPT-A↑
Seen tasks					
Edit Transfer [8]	0.764	0.408	0.241	2.839	2.843
LoRWeB [34]	**0.898**	0.254	0.226	**4.198**	3.033
RelationAdapter [16]	0.858	0.327	0.294	3.201	3.417
Ours-16K	0.896	**0.249**	**0.329**	3.939	**4.234**
Unseen tasks from Relation dataset [16]					
VisualCloze [30]	0.808	0.499	0.286	2.973	3.277
LoRWeB [34]	0.837	0.446	0.270	3.793	3.023
RelationAdapter [16]	0.851	0.423	0.263	3.847	3.432
Ours-16K	0.853	**0.396**	0.351	**4.060**	3.847
Ours-1M	**0.867**	0.399	**0.377**	3.943	**4.077**
Challenging unseen tasks					
VisualCloze [30]	0.827	0.297	0.313	3.416	2.556
LoRWeB [34]	**0.945**	**0.166**	0.203	**4.680**	2.004
RelationAdapter [16]	0.862	0.370	0.305	3.804	2.884
Ours-16K	0.882	0.248	0.382	3.924	3.332
Ours-1M	0.892	0.265	**0.475**	3.964	**4.008**

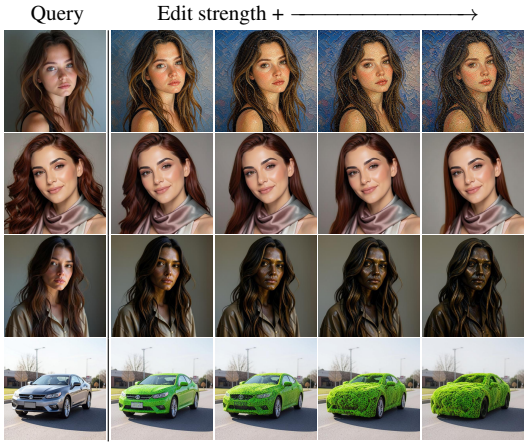


Figure 5: Continuous image editing with Delta-Adapter.

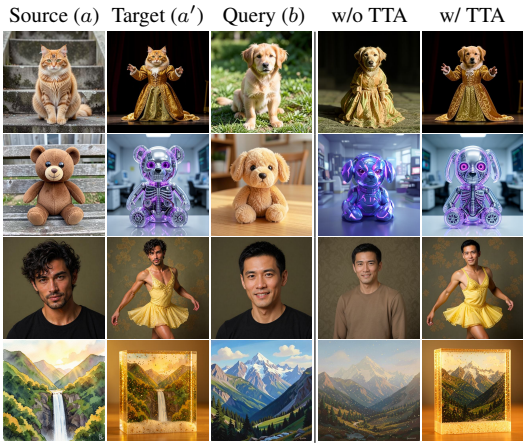


Figure 6: Test-time adaptation (TTA) on challenging unseen exemplar pairs.

Quantitative evaluation. Table 1 presents quantitative results on both seen and unseen tasks. Although LoRWeB achieves the highest consistency scores, its editing accuracy is substantially lower, primarily due to frequent editing failures that leave the output nearly unchanged from the input. This behavior aligns with the qualitative observations in Figures 3 and 4. Among the remaining methods, ours achieves the best editing accuracy and content consistency. On unseen tasks from the Relation dataset [16], Ours-1M achieves a GPT-A score of 4.077, outperforming RelationAdapter's 3.432. This advantage is more pronounced on the more challenging unseen tasks beyond the Relation dataset, where Ours-1M achieves a GPT-A score of 4.008 and a CDS of 0.475, compared with 2.884 and 0.305 for RelationAdapter, respectively. Moreover, scaling the training data from 16K to 1M pairs improves GPT-A from 3.332 to 4.008 on these more challenging tasks, highlighting the benefits of large-scale single-pair supervision for generalization. A human preference study in Appendix I further supports the perceptual quality of our results.

Test-time adaptation. As described in Section 4.5, our single-pair supervision paradigm enables test-time adaptation for challenging unseen exemplar pairs. Empirically, we find that fine-tuning the Delta-Adapter for as few as 20 gradient steps is sufficient. Figure 6 provides qualitative comparisons before and after adaptation on several such difficult cases. Without test-time adaptation, the model captures only the coarse semantics of the intended edit. After adaptation, editing fidelity improves substantially, yielding outputs that faithfully reflect the transformation specified by the exemplar.

5.3 Ablation Study

To validate the contribution of each component in Delta-Adapter, we conduct ablation studies by removing or modifying individual components. Specifically, we evaluate seven variants: (1) *w/o semantic delta*, which uses the full exemplar pair as conditioning instead of the semantic delta; (2) *w/o layernorm*, which removes layer normalization from the semantic delta; (3) *w/o gated residual*, which feeds the normalized delta directly into the resampler without gated residual refinement; (4) *w/o perceiver*, which replaces the Perceiver resampler with a pooling–MLP architecture; (5) *w/o per-token proj.*, which replaces the per-token projection with a shared linear projection; (6) *w/o* $\mathcal{L}_{\mathrm{sdc}}$, which removes the semantic delta consistency loss; and (7) *DINOv2-based* $\mathcal{L}_{\mathrm{sdc}}$, which computes the consistency loss using DINOv2 [39] features while retaining SigLIP features for the conditioning delta. Table 2 reports the quantitative results for all variants, consistently supporting the effectiveness of our design choices. In particular, replacing the Perceiver resampler substantially degrades performance on unseen tasks. The DINOv2-based variant outperforms the variant without the consistency loss, but underperforms our original model with the SigLIP-based consistency loss. This suggests that using a shared semantic feature space for conditioning and consistency regularization provides more coherent supervision. Further analyses of delta conditioning and semantic consistency loss are provided in Appendix B.

Table 2: Ablation study. All variants are trained on the Relation dataset.

Variant	CLIP-I↑	LPIPS↓	GPT-C↑	GPT-A↑
Seen tasks				
w/o semantic delta	0.866	0.337	3.228	3.839
w/o layernorm	0.875	0.276	3.772	3.865
w/o gated residual	0.887	0.273	3.824	4.091
w/o perceiver	0.891	0.260	3.876	4.059
w/o per-token proj.	0.873	0.323	3.568	3.624
w/o $\mathcal{L}_{\mathrm{sdc}}$	0.879	0.287	3.743	4.162
DINOv2-based $\mathcal{L}_{\mathrm{sdc}}$	0.890	0.271	3.867	4.196
Ours	**0.896**	**0.249**	**3.939**	**4.234**
Unseen tasks				
w/o perceiver	0.881	0.285	3.828	2.964
Ours	**0.882**	**0.248**	**3.924**	**3.332**

6 Conclusions and Limitations

We presented Delta-Adapter, a framework for exemplar-based image editing under single-pair supervision without textual guidance. By conditioning on a semantic delta rather than the edited image directly, our method eliminates the need for paired exemplars at training time, enabling scalable data curation and effective test-time adaptation. Experiments demonstrate superior performance over four strong baselines on both seen and unseen editing tasks. One limitation of our approach is that the semantic delta is bounded by the representational capacity of the vision encoder, which operates at a high semantic level and may fail to capture fine-grained visual details such as text rendered within images. Several failure examples are provided in Appendix G. Additionally, our model is currently trained on approximately one million image pairs; we anticipate that scaling to larger datasets will further improve editing generalization, which we leave as future work.

Acknowledgments and Disclosures

This work is supported by National Natural Science Foundation of China (No. 62302535 and No. 62176223), Guangdong Basic and Applied Basic Research Foundation (No. 2026A1515011830), and Zhuhai Basic and Applied Basic Research Foundation (No. 2320004002745). The authors declare no competing interests.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.

[2] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *CVPR*, 2022.

[3] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. In *SIGGRAPH*, 2023.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei A. Efros. Visual prompting via image inpainting. In *NeurIPS*, 2022.

[6] Manuel Brack, Felix Friedrich, Katharina Kornmeier, Linoy Tsaban, Patrick Schramowski, Kristian Kersting, and Apolinário Passos. Ledits++: Limitless image editing using text-to-image models. In *CVPR*, 2024.

[7] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, 2023.

[8] Lan Chen, Qi Mao, Yuchao Gu, and Mike Zheng Shou. Edit transfer: Learning image editing via vision in-context relations. *arXiv preprint arXiv:2503.13327*, 2025.

[9] Shuo Chen, Fengming Huang, Yu Yao, Mingming Gong, and Tongliang Liu. Scalable minimal-change learning for controllable image editing. *arXiv preprint arXiv:2610.06021*, 2026.

[10] Xi Chen, Lianghua Huang, Yu Liu, Yujun Shen, Deli Zhao, and Hengshuang Zhao. Anydoor: Zero-shot object-level image customization. In *CVPR*, 2024.

[11] Riccardo Corvi, Davide Cozzolino, Giada Zingarini, Giovanni Poggi, Koki Nagano, and Luisa Verdoliva. On the detection of synthetic images generated by diffusion models. In *ICASSP*, 2023.

[12] Guillaume Couairon, Jakob Verbeek, Holger Schwenk, and Matthieu Cord. Diffedit: Diffusion-based semantic image editing with mask guidance. In *ICLR*, 2023.

[13] Google DeepMind. Nano banana 2, 2026.

[14] Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. Guiding instruction-based image editing via multimodal large language models. In *ICLR*, 2024.

[15] Zigang Geng, Binxin Yang, Tiankai Hang, Chen Li, Shuyang Gu, Ting Zhang, Jianmin Bao, Zheng Zhang, Han Hu, Dong Chen, and Baining Guo. Instructdiffusion: A generalist modeling interface for vision tasks. In *CVPR*, 2024.

[16] Yan Gong, Yiren Song, Yicheng Li, Chenglin Li, and Yin Zhang. Relationadapter: Learning and transferring visual relation with diffusion transformers. In *NeurIPS*, 2025.

[17] Zheng Gu, Shiyuan Yang, Jing Liao, Jing Huo, and Yang Gao. Analogist: Out-of-the-box visual in-context learning with image diffusion model. In *SIGGRAPH*, 2024.

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

[19] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *SIGGRAPH*, 2001.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[21] Yuzhou Huang, Liangbin Xie, Xintao Wang, Ziyang Yuan, Xiaodong Cun, Yixiao Ge, Jiantao Zhou, Chao Dong, Rui Huang, Ruimao Zhang, and Ying Shan. Smartedit: Exploring complex instruction-based image editing with multimodal large language models. In *CVPR*, 2024.

[22] Saar Huberman, Or Patashnik, Omer Dahary, Ron Mokady, and Daniel Cohen-Or. Image generation from contextually-contradictory prompts. *arXiv preprint arXiv:2506.01929*, 2025.

[23] Maxwell Jones, Sheng-Yu Wang, Nupur Kumari, David Bau, and Jun-Yan Zhu. Customizing text-to-image models with a single image pair. In *SIGGRAPH Asia*, 2024.

[24] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *CVPR*, 2023.

[25] Gwanghyun Kim, Taesung Kwon, and Jong Chul Ye. Diffusionclip: Text-guided diffusion models for robust image manipulation. In *CVPR*, 2022.

[26] Maksim Kuprashevich, Grigorii Alekseenko, Irina Tolstykh, Georgii Fedorov, Bulat Suleimanov, Vladimir Dokholyan, and Aleksandr Gordeev. Nohumansrequired: Autonomous high-quality image editing triplet mining. In *WACV*, 2026.

[27] Black Forest Labs. Flux, 2024.

[28] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, Sumith Kulal, Kyle Lacey, Yam Levi, Cheng Li, Dominik Lorenz, Jonas Müller, Dustin Podell, Robin Rombach, Harry Saini, Axel Sauer, and Luke Smith. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

[29] Dongxu Li, Junnan Li, and Steven C. H. Hoi. Blip-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing. *arXiv preprint arXiv:2305.14720*, 2023.

[30] Zhong-Yu Li, Ruoyi Du, Juncheng Yan, Le Zhuo, Zhen Li, Peng Gao, Zhanyu Ma, and Ming-Ming Cheng. Visualcloze: A universal image generation framework via visual in-context learning. In *ICCV*, 2025.

[31] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

[32] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023.

[33] Haoguang Lu, Jiacheng Chen, Zhenguo Yang, Aurele Tohokantche Gnanha, Fu Lee Wang, Li Qing, and Xudong Mao. Pairedit: Learning semantic variations for exemplar-based image editing. In *NeurIPS*, 2025.

[34] Hila Manor, Rinon Gal, Haggai Maron, Tomer Michaeli, and Gal Chechik. Spanning the visual analogy space with a weight basis of loras. *arXiv preprint arXiv:2602.15727*, 2026.

[35] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022.

[36] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *arXiv preprint arXiv:2302.08453*, 2023.

[37] Thao Nguyen, Yuheng Li, Utkarsh Ojha, and Yong Jae Lee. Visual instruction inversion: Image editing via visual prompting. In *NeurIPS*, 2023.

[38] OpenAI. Gpt-image-2, 2026.

[39] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[40] Gaurav Parmar, Krishna Kumar Singh, Richard Zhang, Yijun Li, Jingwan Lu, and Jun-Yan Zhu. Zero-shot image-to-image translation. In *SIGGRAPH*, 2023.

[41] Or Patashnik, Daniel Garibi, Idan Azuri, Hadar Averbuch-Elor, and Daniel Cohen-Or. Localizing object-level shape variations with text-to-image diffusion models. In *ICCV*, 2023.

[42] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[43] Yusu Qian, Eli Bocek-Rivele, Liangchen Song, Jialing Tong, Yinfei Yang, Jiasen Lu, Wenze Hu, and Zhe Gan. Pico-banana-400k: A large-scale dataset for text-guided image editing. *arXiv preprint arXiv:2510.19808*, 2025.

[44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[46] Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. In *CVPR*, 2024.

[47] Xue Song, Jiequan Cui, Hanwang Zhang, Jiaxin Shi, Jingjing Chen, Chi Zhang, and Yu-Gang Jiang. Lora of change: Learning to generate lora for the editing instruction from a single before-after image pair. *arXiv preprint arXiv:2411.19156*, 2024.

[48] Ashutosh Srivastava, Tarun Ram Menta, Abhinav Java, Avadhoot Gorakh Jadhav, Silky Singh, Surgan Jandial, and Balaji Krishnamurthy. Reedit: Multimodal exemplar-based image editing. In *WACV*, 2025.

[49] Yasheng Sun, Yifan Yang, Houwen Peng, Yifei Shen, Yuqing Yang, Han Hu, Lili Qiu, and Hideki Koike. Imagebrush: Learning visual in-context instructions for exemplar-based image manipulation. In *NeurIPS*, 2023.

[50] Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *CVPR*, 2023.

[51] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A. Efros. Cnn-generated images are surprisingly easy to spot... for now. In *CVPR*, 2020.

[52] Su Wang, Chitwan Saharia, Ceslee Montgomery, Jordi Pont-Tuset, Shai Noy, Stefano Pellegrini, Yasumasa Onoe, Sarah Laszlo, David J. Fleet, Radu Soricut, Jason Baldridge, Mohammad Norouzi, Peter Anderson, and William Chan. Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In *CVPR*, 2023.

[53] Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In *CVPR*, 2023.

[54] Zhendong Wang, Yifan Jiang, Yadong Lu, Yelong Shen, Pengcheng He, Weizhu Chen, Zhangyang Wang, and Mingyuan Zhou. In-context learning unlocked for diffusion models. *arXiv preprint arXiv:2305.01115*, 2023.

[55] Zeren Xiong, Yue Yu, Zedong Zhang, Shuo Chen, Jian Yang, and Jun Li. Vmdiff: Visual mixing diffusion for limitless cross-object synthesis. *arXiv preprint arXiv:2509.23605*, 2025.

[56] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models. In *CVPR*, 2023.

[57] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

[58] Tao Yu, Runseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting. *arXiv preprint arXiv:2304.06790*, 2023.

[59] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023.

[60] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

[61] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[62] Shu Zhang, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, Caiming Xiong, and Ran Xu. Hive: Harnessing human feedback for instructional visual editing. In *CVPR*, 2024.

[63] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. What makes good examples for visual in-context learning? *arXiv preprint arXiv:2301.13670*, 2023.

[64] Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K. Wong. Uni-controlnet: All-in-one control to text-to-image diffusion models. *arXiv preprint arXiv:2305.16322*, 2023.

[65] Junhao Zhuang, Yanhong Zeng, Wenran Liu, Chun Yuan, and Kai Chen. A task is worth one word: Learning with task prompts for high-quality versatile image inpainting. In *ECCV*, 2024.

A Implementation Details

Our method. We build Delta-Adapter on top of FLUX.2-klein-4B[2], using SigLIP-2 [59] (`google/siglip2-base-patch16-224`) as the image encoder. The mapping layers consist of a Perceiver resampler [1] with 128 learnable queries. We train with the AdamW optimizer (lr $= 1 \times 10^{-4}$, weight decay $= 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.99$) in bfloat16 precision. The full model is trained on approximately one million image pairs for 100K steps, with a per-GPU batch size of 16 across $4\times$ H200 GPUs. For the seen-task comparisons in Figure 3 and Table 1, we additionally train a model exclusively on 16K image pairs from the Relation dataset to ensure a fair comparison with RelationAdapter and LoRWeB, using a batch size of 32 for 10K steps. Throughout training, only the mapping layers and the newly introduced cross-attention layers are optimized, while keeping FLUX.2-klein and SigLIP-2 frozen. For test-time adaptation, we fine-tune Delta-Adapter for 20 gradient steps with a batch size of 1 and a learning rate of 1×10^{-4}. Notably, this fine-tuning requires only 33 GB of GPU memory, enabling practical deployment on a single GPU.

Baselines. For RelationAdapter [16], LoRWeB [34], and VisualCloze [30], we evaluate using their official pre-trained checkpoints and default inference configurations. For Edit Transfer [8], we follow the official training protocol and retrain the model on the Relation dataset to ensure a fair comparison. As these baselines additionally require a text instruction alongside the exemplar pair, we generate a concise edit description for each evaluation case using GPT-5.4, conditioned on the exemplar pair $a \rightarrow a'$. All methods are evaluated on identical query images and exemplar pairs under a fixed random seed of 42 to ensure reproducibility.

B Additional Ablation Study

Controlled backbone comparison. To control for backbone and projection layer differences, we retrain our method on FLUX.1-Kontext, which is fine-tuned from the same FLUX.1 model used by RelationAdapter [16]. We also adopt the same projection layer as RelationAdapter and train on the same dataset. As shown in Table 3, our method still substantially outperforms RelationAdapter in both editing accuracy and content consistency, further supporting the effectiveness of delta conditioning.

Table 3: **Controlled comparison on FLUX.1-Kontext.** Ours-16K (Kontext) uses a FLUX.1-Kontext backbone derived from the FLUX.1 model used by RelationAdapter [16], with the same projection layer and training dataset.

Method	CLIP-I↑	LPIPS↓	CDS↑	GPT-C↑	GPT-A↑
RelationAdapter [16]	0.858	0.327	0.294	3.201	3.417
Ours-16K (Kontext)	**0.892**	**0.281**	**0.324**	**3.916**	**4.038**

Semantic consistency loss. The benefit of $\mathcal{L}_{\text{sdc}}$ is relatively modest when averaged over all 218 seen tasks, as many simpler transformations can already be learned effectively with the flow-matching objective alone. To further investigate its effectiveness, we evaluate the ablation on a subset of 40 challenging seen tasks (e.g., gesture editing). As shown in Table 4, removing $\mathcal{L}_{\text{sdc}}$ decreases GPT-A from 4.380 to 3.785, demonstrating that semantic delta consistency is particularly beneficial for challenging editing transformations.

Table 4: **Ablation of the semantic consistency loss on 40 challenging seen tasks.**

Method	CLIP-I↑	LPIPS↓	GPT-C↑	GPT-A↑
w/o $\mathcal{L}_{\text{sdc}}$	0.873	0.373	3.280	3.785
Ours	**0.920**	**0.271**	**3.875**	**4.380**

[2] `https://huggingface.co/black-forest-labs/FLUX.2-klein-4B`

C Additional Qualitative Results

Figures 9–15 present additional qualitative results of Delta-Adapter on editing tasks from the Relation dataset [16]. Figure 16 shows results on a broader range of challenging image editing tasks. We also provide additional qualitative comparisons with baseline methods on seen and unseen editing tasks in Figures 17 and 18, respectively.

D Cross-Domain Generalization

To further assess the cross-domain generalization of Delta-Adapter, we evaluate five editing transformations from the Relation dataset [16]: adding a hat, making the subject chubbier, closing the eyes, shouting, and smiling. We use exemplar pairs from the human domain to guide the corresponding edits on animal-domain query images. As shown in Figure 8, Delta-Adapter successfully transfers the intended edits across this substantial domain shift while preserving the identity and structure of the query subjects. In contrast, RelationAdapter transfers some edits but often introduces unintended changes to the subjects' appearance or structure, or fails to accurately apply the demonstrated transformation.

E Single-Pair Supervision of Existing Methods

To investigate whether existing exemplar-based editing methods can learn effectively under single-pair supervision, we retrain Edit Transfer [8] and LoRWeB [34] using individual source-target image pairs. As shown in Figure 7, Edit Transfer tends to copy the exemplar target while largely ignoring the query image. LoRWeB also exhibits substantial leakage of exemplar content, struggling to separate the intended editing semantics from exemplar-specific visual details. In contrast, Delta-Adapter transfers the intended transformations while preserving the identity and structure of the query subjects. These failure modes underscore the challenge of learning transferable editing semantics from single-pair supervision and further motivate our delta-conditioning formulation.

F Comparison with Additional Baselines

In Figure 19, we compare Delta-Adapter against two state-of-the-art multimodal generation models, Nano Banana 2 [13] and GPT-Image-2 [38]. For both models, we provide the exemplar source, exemplar target, and query images, along with the prompt: *"Analyze the visual transformations from Image 1 to Image 2, then apply the same transformations consistently to Image 3 while preserving its original subject, identity, composition, and essential details."* As general-purpose generation systems, both models exhibit notable limitations in editing accuracy and content consistency when applied to exemplar-based image editing. Specifically, they often fail to capture the intended edits (rows 1–3) and tend to leak appearance cues from the exemplar images into the output (rows 4–6). In contrast, Delta-Adapter consistently transfers the target edit while faithfully preserving the content of the query image. In Figure 20, we further compare against PairEdit [33], an optimization-based method that performs test-time optimization independently for each exemplar pair. PairEdit struggles to capture complex edits from the exemplar, whereas Delta-Adapter faithfully transfers the intended transformations to the query image.

G Failure Cases

While Delta-Adapter generalizes well across a wide range of editing tasks, it can struggle to preserve fine-grained visual details, particularly textual content. Figure 21 illustrates representative failure cases where the model generates characters inconsistent with those in the exemplar pair. This limitation is partly due to the pre-trained vision encoder, which may not faithfully capture such fine-grained visual details.

Table 5: Quantitative results using Claude Opus 4.8 as the evaluator in place of GPT-5.4.

Method	VLM-C↑	VLM-A↑
Seen tasks		
Edit Transfer [8]	2.792	2.575
LoRWeB [34]	**4.083**	2.798
RelationAdapter [16]	3.488	3.397
Ours-16K	3.964	**4.025**
Challenging unseen tasks		
VisualCloze [30]	3.208	2.412
LoRWeB [34]	**4.696**	1.928
RelationAdapter [16]	3.852	2.584
Ours-16K	3.960	3.156
Ours-1M	3.984	**3.872**

Table 6: Quantitative results using GPT-5.4 with the evaluation prompt from LoRWeB [34].

Method	GPT-C↑	GPT-A↑
Seen tasks		
Edit Transfer [8]	2.650	2.694
LoRWeB [34]	**4.187**	3.023
RelationAdapter [16]	3.477	3.531
Ours-16K	4.019	**4.183**
Challenging unseen tasks		
VisualCloze [30]	3.300	2.296
LoRWeB [34]	**4.704**	2.012
RelationAdapter [16]	3.872	2.624
Ours-16K	3.944	3.192
Ours-1M	3.988	**3.936**

H Details of GPT-Based Evaluation Metrics

We employ GPT-5.4 as an automated evaluator to assess each edited result along two dimensions: editing accuracy (GPT-A) and content consistency (GPT-C). For each instance, the evaluator receives four images as input: the reference source a, the reference target a', the query image b, and the candidate edit $\hat{b}'$. It is then prompted to judge whether the intended transformation $a \to a'$ is faithfully applied to b, and whether the non-edited content of the query image is preserved. The complete system prompt is provided in Figure 22.

Robustness to evaluator and prompt choice. To assess whether our conclusions are sensitive to the choice of evaluator or prompt, we conduct two additional evaluations: (i) replacing GPT-5.4 with Claude Opus 4.8 while keeping the evaluation protocol unchanged, and (ii) adopting the evaluation prompt used by LoRWeB [34]. As shown in Tables 5 and 6, Delta-Adapter achieves the highest editing accuracy on both seen and unseen tasks in both settings. Although LoRWeB obtains higher content consistency scores, this advantage largely stems from frequent editing failures that leave the output nearly unchanged from the query image.

Reliability of GPT-based evaluation. To assess the reliability of GPT-based evaluation, we randomly sample 100 examples and collect independent ratings from six human annotators using the same evaluation rubric. We compute the mean Pearson correlation between GPT scores and individual human ratings, as well as the mean pairwise correlation among human annotators. GPT-A and GPT-C show strong correlations with human ratings ($r = 0.869$, 95% CI $[0.790, 0.920]$ and $r = 0.772$, 95% CI $[0.628, 0.861]$, respectively), closely matching the corresponding inter-human correlations ($r = 0.888$, 95% CI $[0.844, 0.920]$ and $r = 0.780$, 95% CI $[0.684, 0.848]$, respectively).

I Human Preference Evaluation

To further validate the perceptual quality of our results, we conduct a pairwise preference study comparing Delta-Adapter against each baseline. In each trial, participants are presented with an exemplar pair and a query image, alongside two candidate edits: one produced by our method and one by a baseline. They are asked to select the edit that better captures the transformation demonstrated by the exemplar pair while preserving the unedited regions of the query image. We collect 768 judgments from 32 participants, with responses evenly distributed across seen and unseen editing tasks. As reported in Table 7, participants consistently preferred Delta-Adapter over all baselines by a substantial margin. A representative interface screenshot is shown in Figure 24.

Table 7: Pairwise user preference study comparing our method with each baseline.

Baseline	Pref. Baseline	Pref. Ours
VisualCloze [30]	13.54 %	**86.46 %**
Edit Transfer [8]	6.77 %	**93.23 %**
LoRWeB [34]	10.42 %	**89.58 %**
RelationAdapter [16]	19.27 %	**80.73 %**

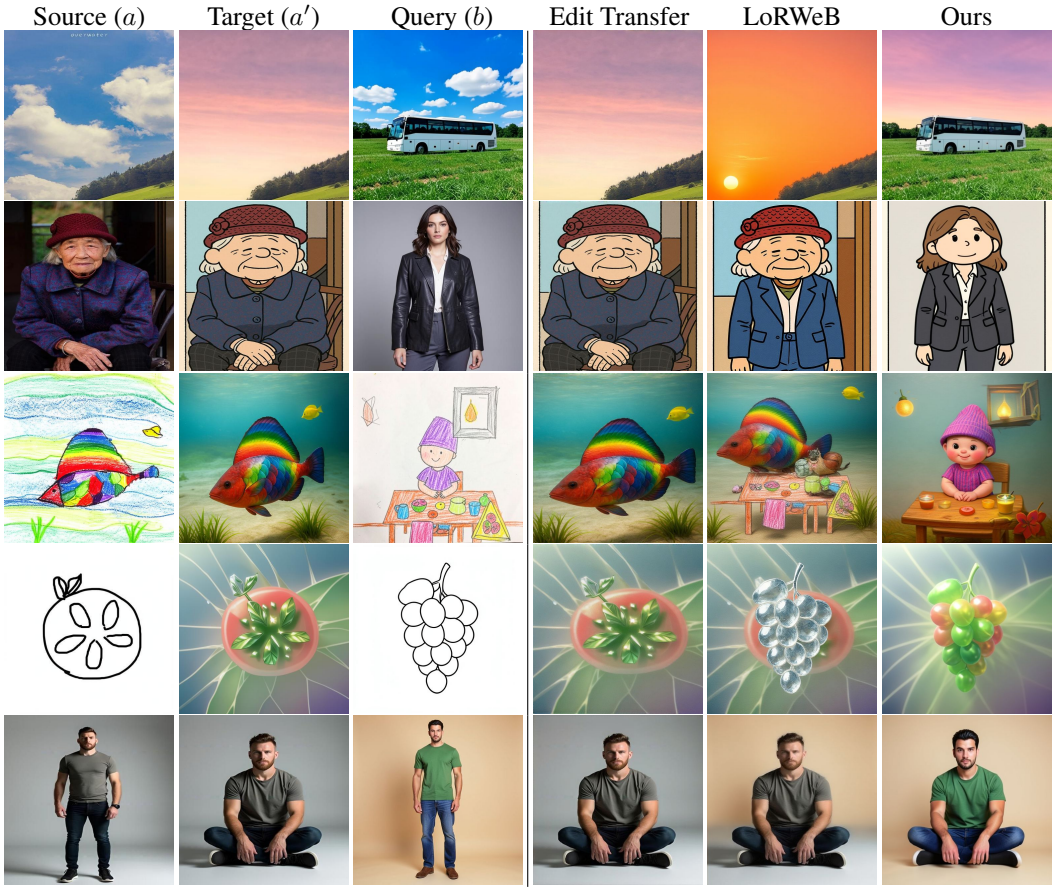


Figure 7: **Comparison under single-pair supervision.** Edit Transfer [8] and LoRWeB [34] are retrained using individual source-target pairs. Edit Transfer tends to copy the exemplar target while ignoring the query, whereas LoRWeB suffers from exemplar-content leakage, introducing unwanted appearance or objects from the exemplar into the output. In contrast, Delta-Adapter successfully transfers the intended transformations while preserving the query content.

J Societal Impact

Delta-Adapter enables image edits specified entirely through visual exemplar pairs, without textual instructions, lowering the barrier to exemplar-based editing and supporting applications such as content creation, artistic exploration, and visual prototyping. Like other generative editing methods, it may be misused to create misleading imagery. Mitigating such risks calls for reliable detection of synthetic or edited images [51, 11], along with responsible deployment practices such as provenance tracking and user-facing disclosures.

K Licenses for Pre-trained Models and Datasets

Our implementation builds upon the publicly available FLUX.2-klein-4B model, released under the Apache 2.0 License, and the SigLIP-2 image encoder, released under the Apache 2.0 License. For training, we use three publicly available image-pair datasets: Relation [16], Pico-Banana [43], and NHR-Edit [26], and adhere to the licenses specified by their respective authors. All evaluation images are drawn from these datasets or sourced from [55, 22, 60, 33].

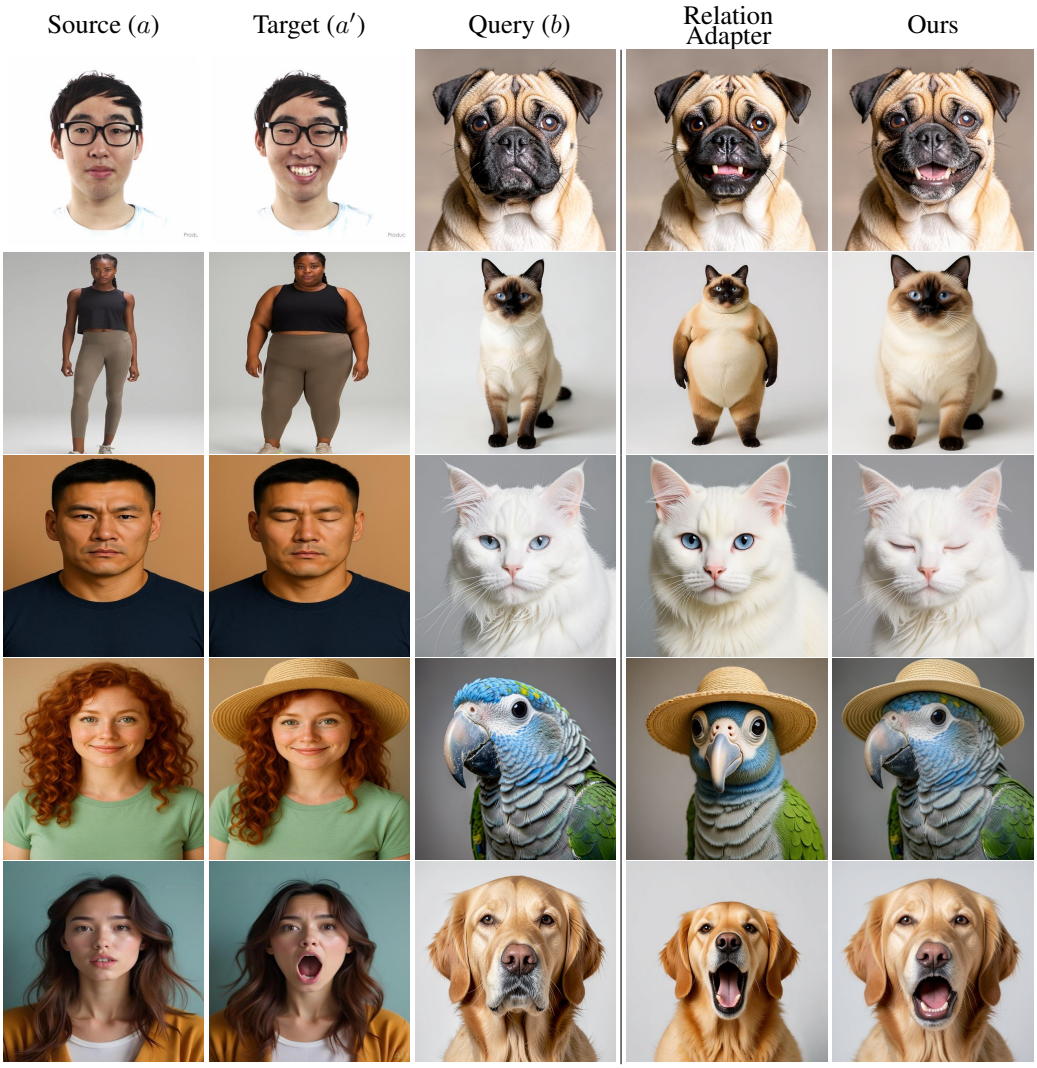


Figure 8: **Cross-domain generalization from human to animal images.** Qualitative comparison between Delta-Adapter and RelationAdapter [16] on five editing transformations: smiling, becoming chubbier, closing eyes, wearing a hat, and shouting (from top to bottom). Given human-domain exemplar pairs, Delta-Adapter more faithfully transfers the demonstrated transformations to animal-domain query images while preserving their identity and structure.

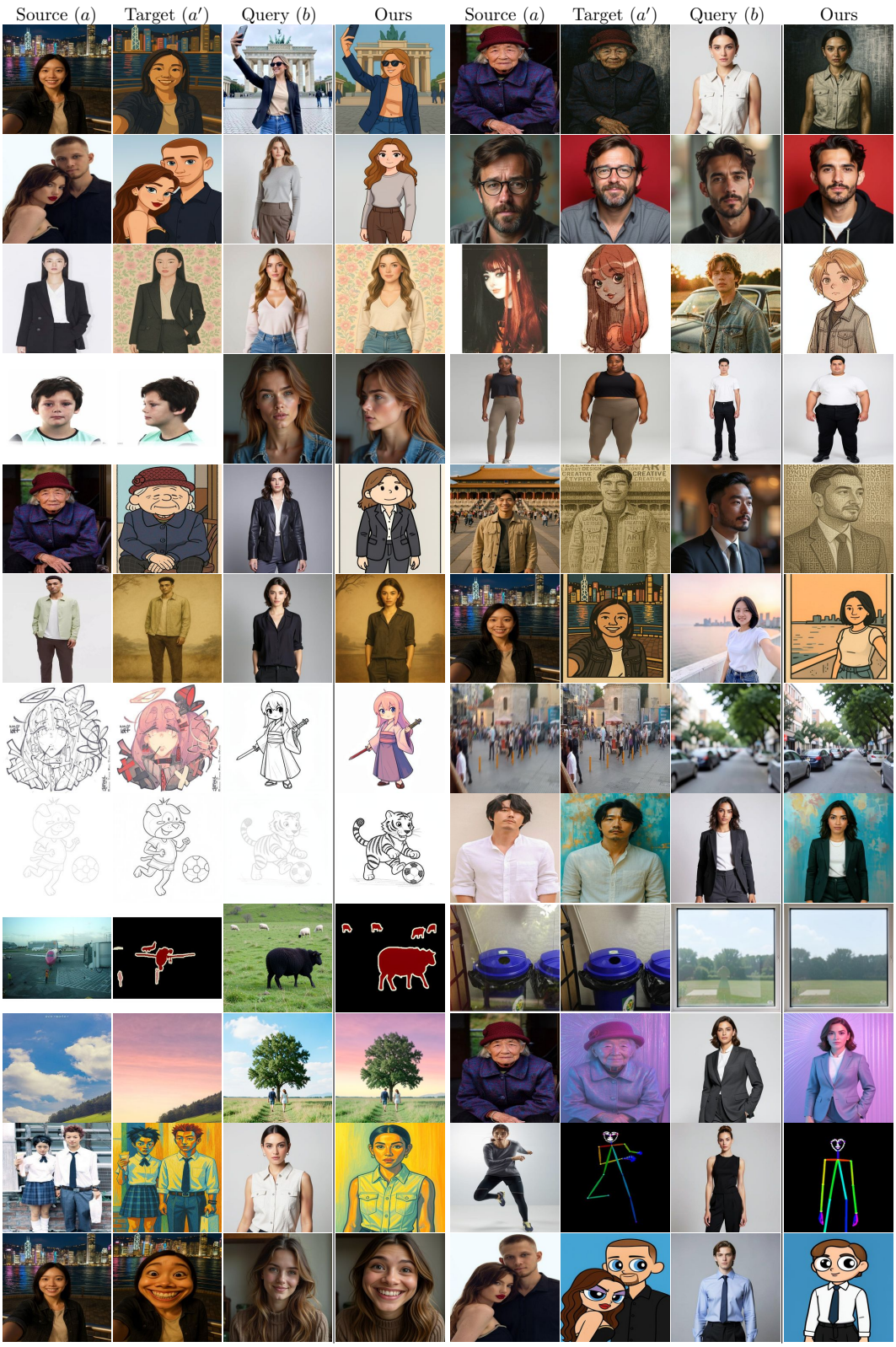


Figure 9: **Additional editing results produced by Delta-Adapter on the Relation dataset [16].** Given an exemplar pair, our method infers the underlying transformation and faithfully applies it to an unseen query image.

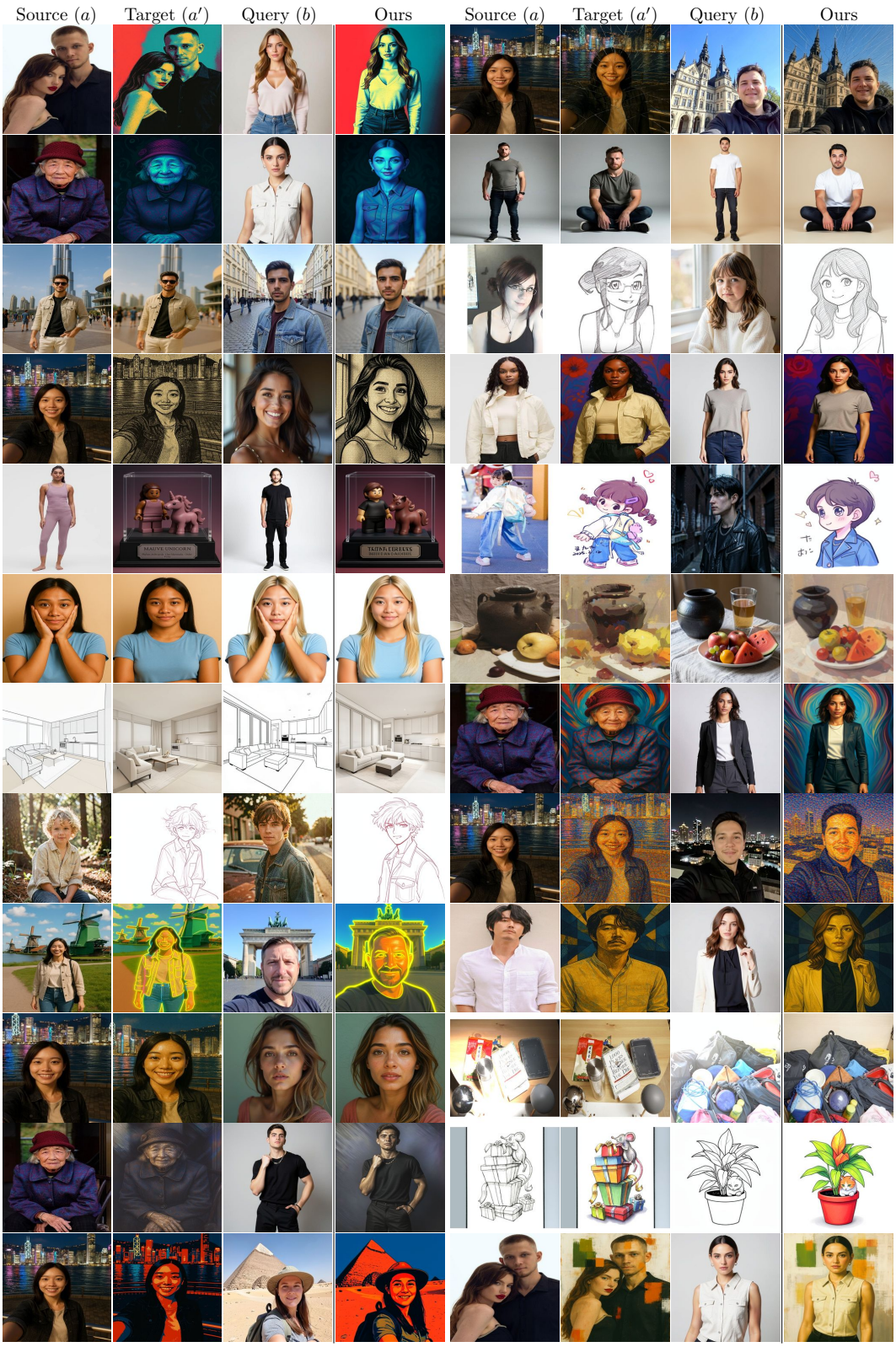


Figure 10: **Additional editing results produced by Delta-Adapter on the Relation dataset [16].** Given an exemplar pair, our method infers the underlying transformation and faithfully applies it to an unseen query image.

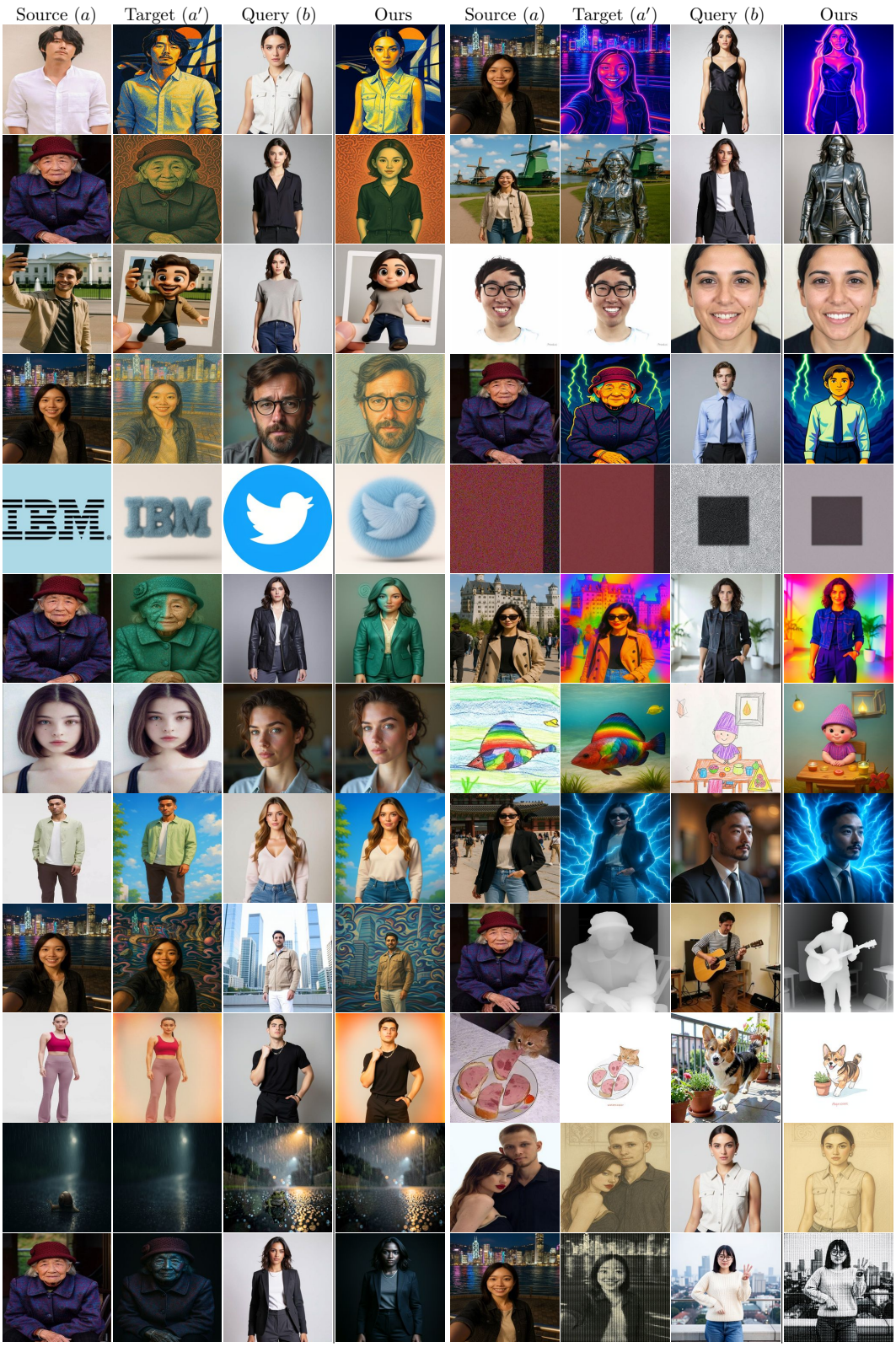


Figure 11: **Additional editing results produced by Delta-Adapter on the Relation dataset [16].** Given an exemplar pair, our method infers the underlying transformation and faithfully applies it to an unseen query image.



Figure 12: **Additional editing results produced by Delta-Adapter on the Relation dataset [16].** Given an exemplar pair, our method infers the underlying transformation and faithfully applies it to an unseen query image.

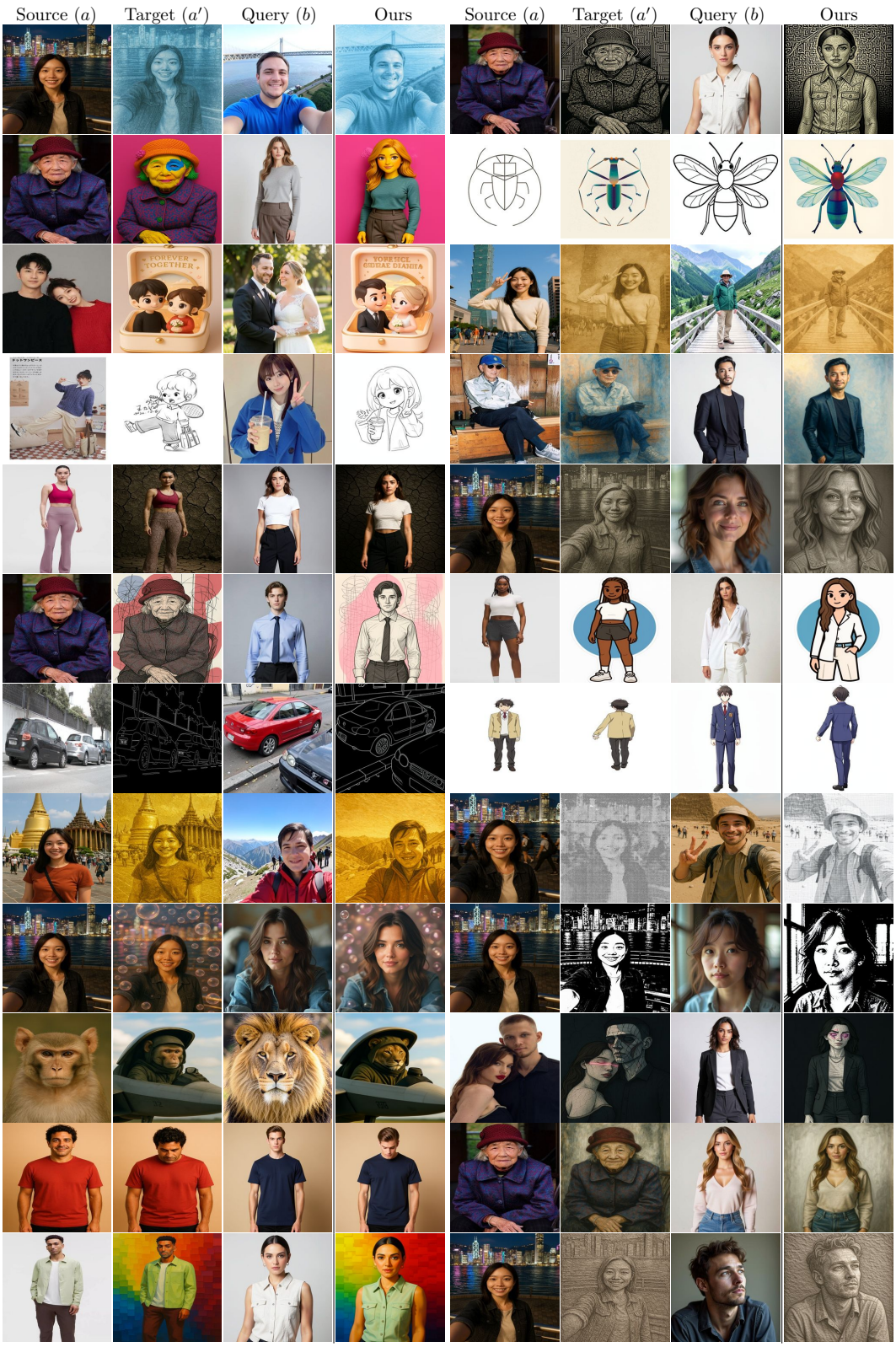


Figure 13: **Additional editing results produced by Delta-Adapter on the Relation dataset [16].** Given an exemplar pair, our method infers the underlying transformation and faithfully applies it to an unseen query image.

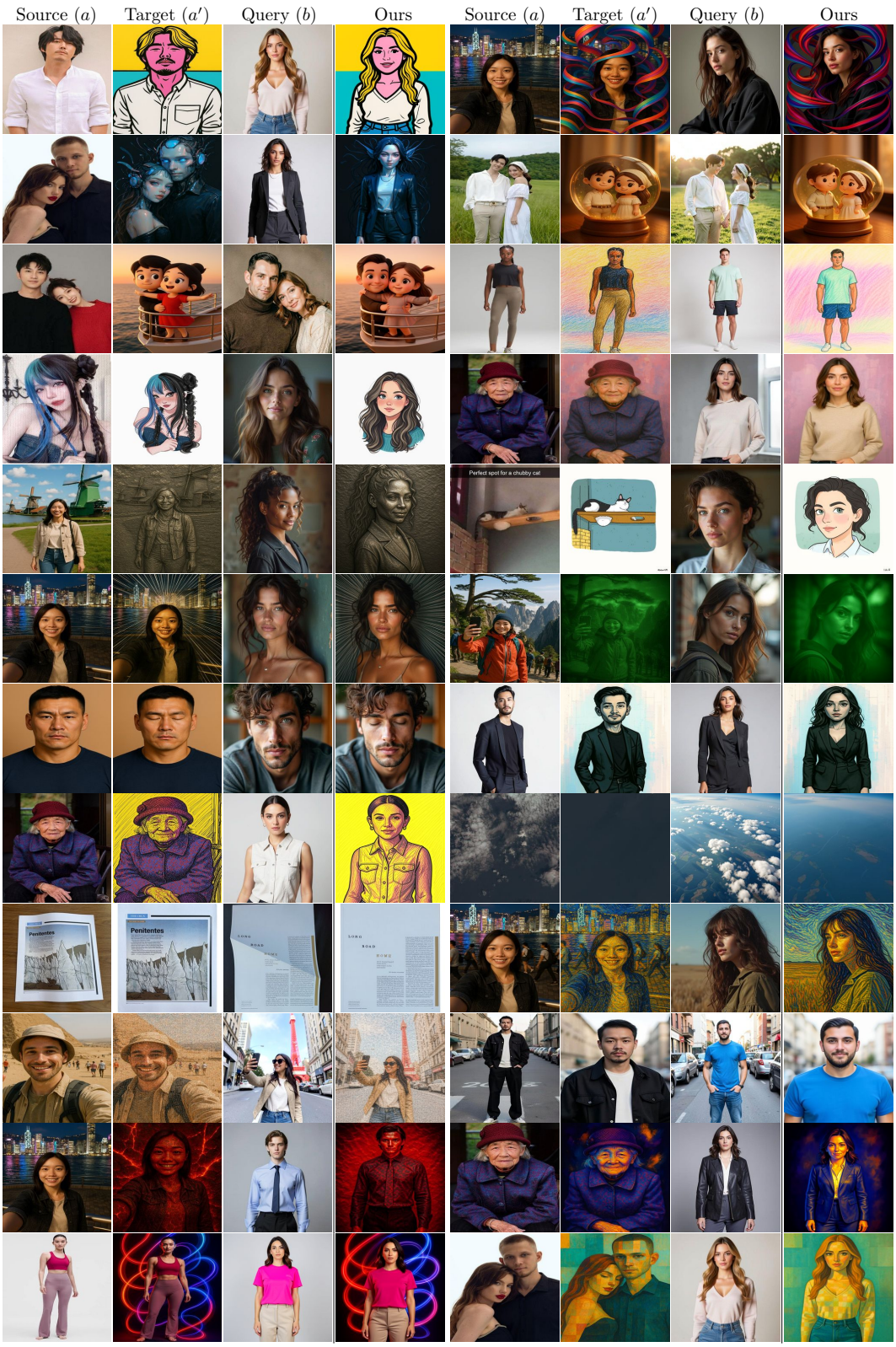


Figure 14: **Additional editing results produced by Delta-Adapter on the Relation dataset [16].** Given an exemplar pair, our method infers the underlying transformation and faithfully applies it to an unseen query image.

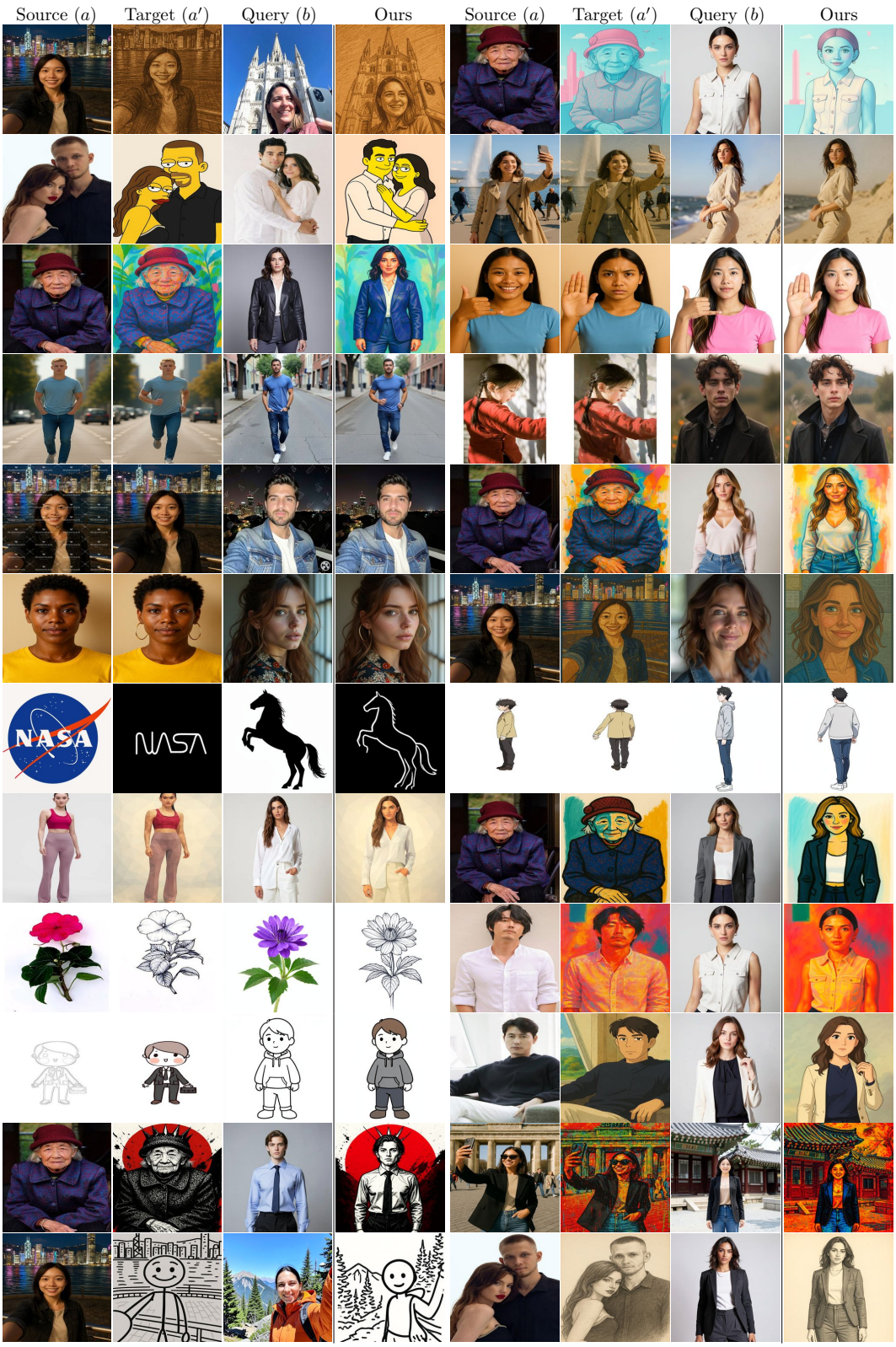


Figure 15: **Additional editing results produced by Delta-Adapter on the Relation dataset [16].** Given an exemplar pair, our method infers the underlying transformation and faithfully applies it to an unseen query image.

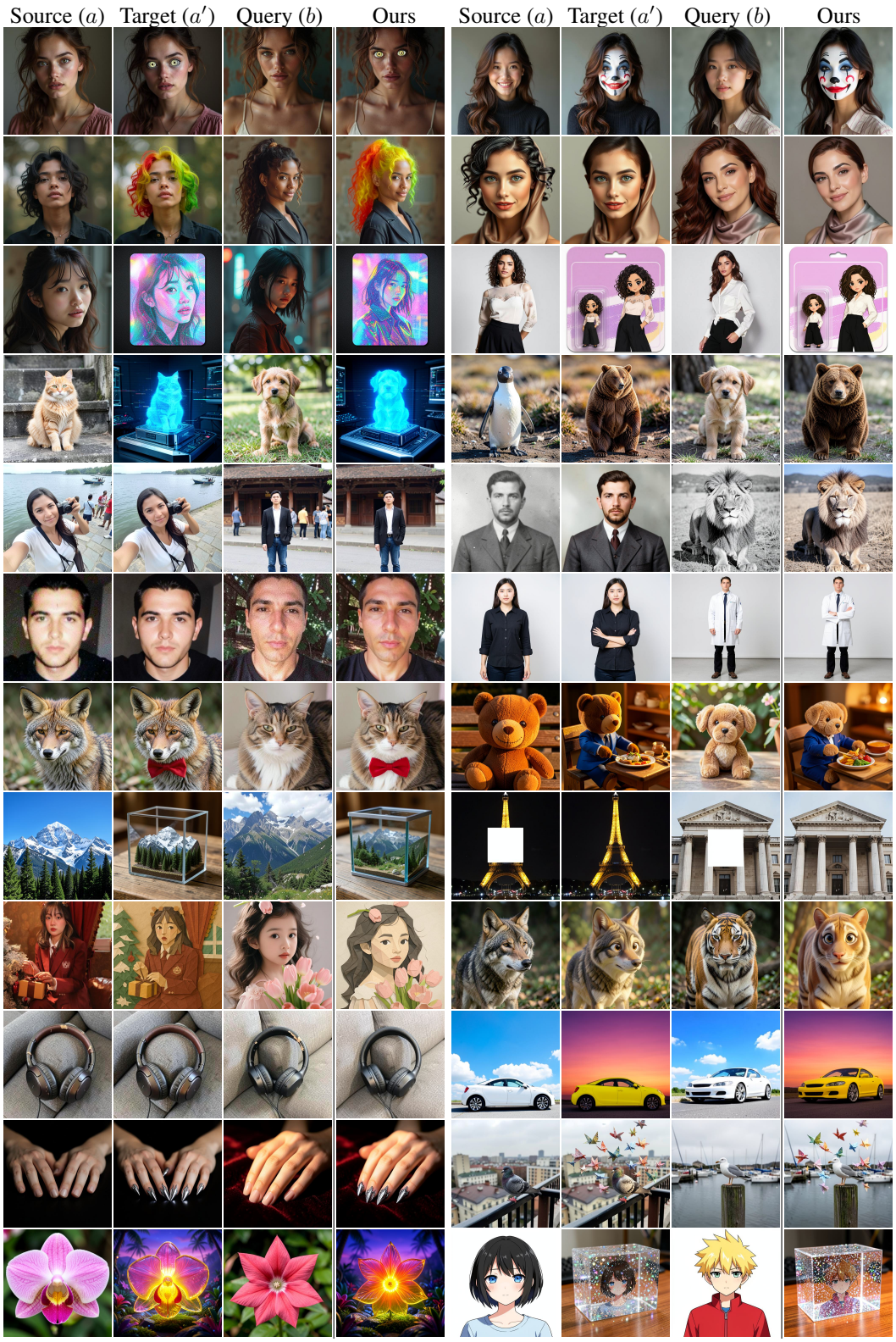


Figure 16: **Additional editing results produced by Delta-Adapter.** Given an exemplar pair, our method infers the underlying transformation and faithfully applies it to unseen query images.

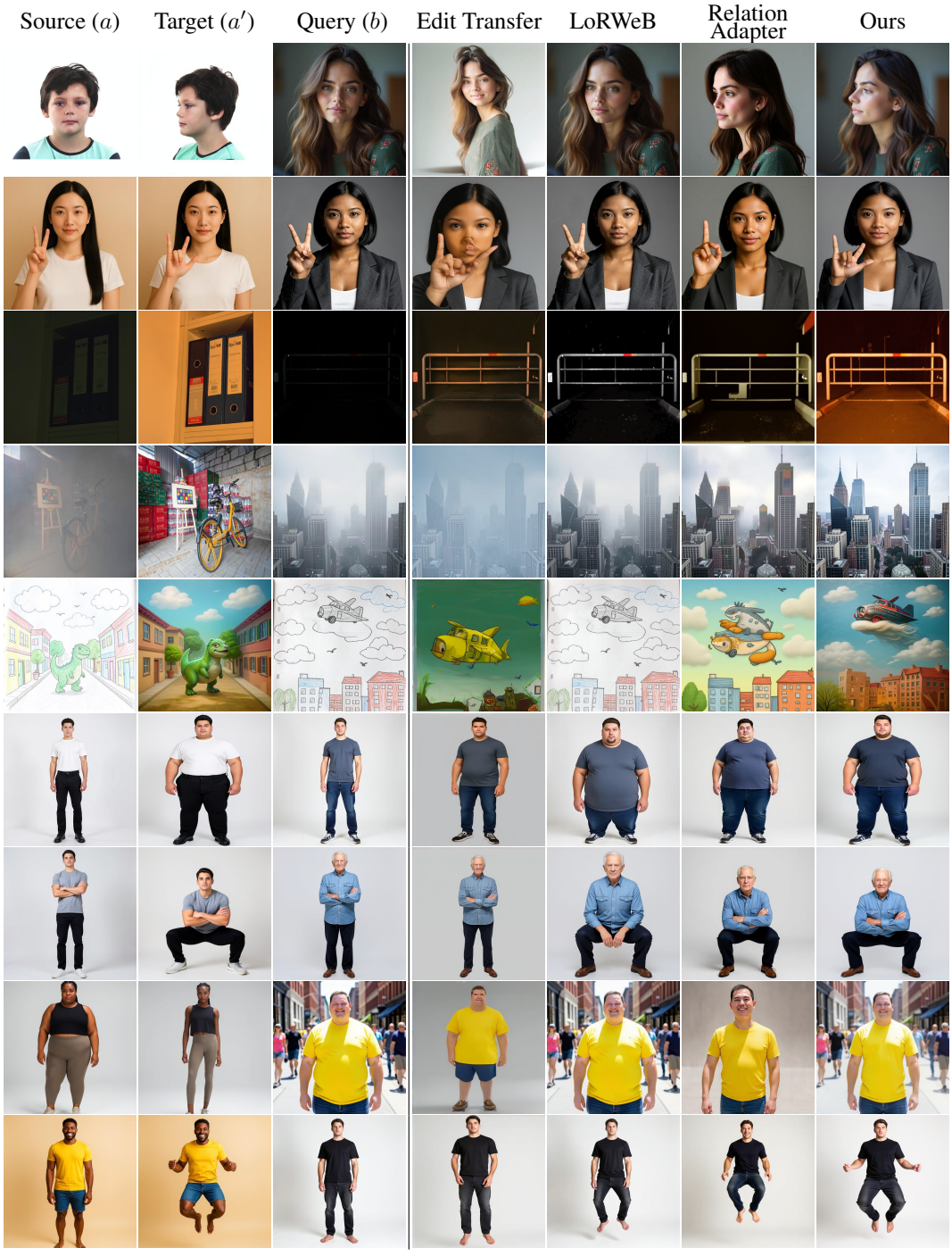


Figure 17: **Additional qualitative comparison on seen editing tasks.** We compare Delta-Adapter with RelationAdapter [16], LoRWeB [34], and Edit Transfer [8]. Delta-Adapter more faithfully captures the edit semantics implied by the exemplar pair and applies them to the query image, while better preserving its underlying structure and identity.

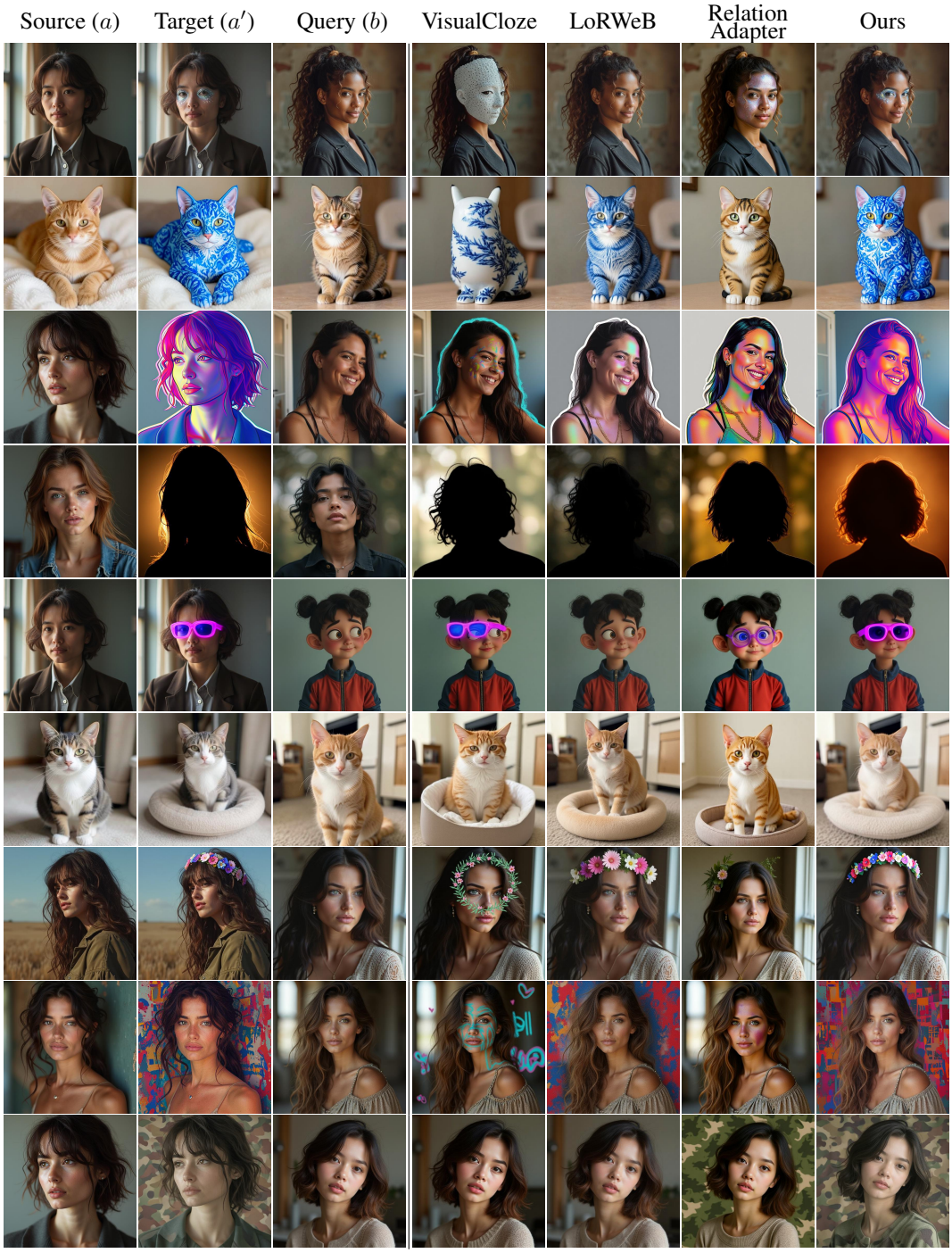


Figure 18: **Additional qualitative comparison on unseen editing tasks.** We compare Delta-Adapter with RelationAdapter [16], LoRWeB [34], and VisualCloze [30]. Across diverse unseen transformations, Delta-Adapter produces outputs that are semantically aligned with the exemplar edit, demonstrating superior generalization over the baselines.

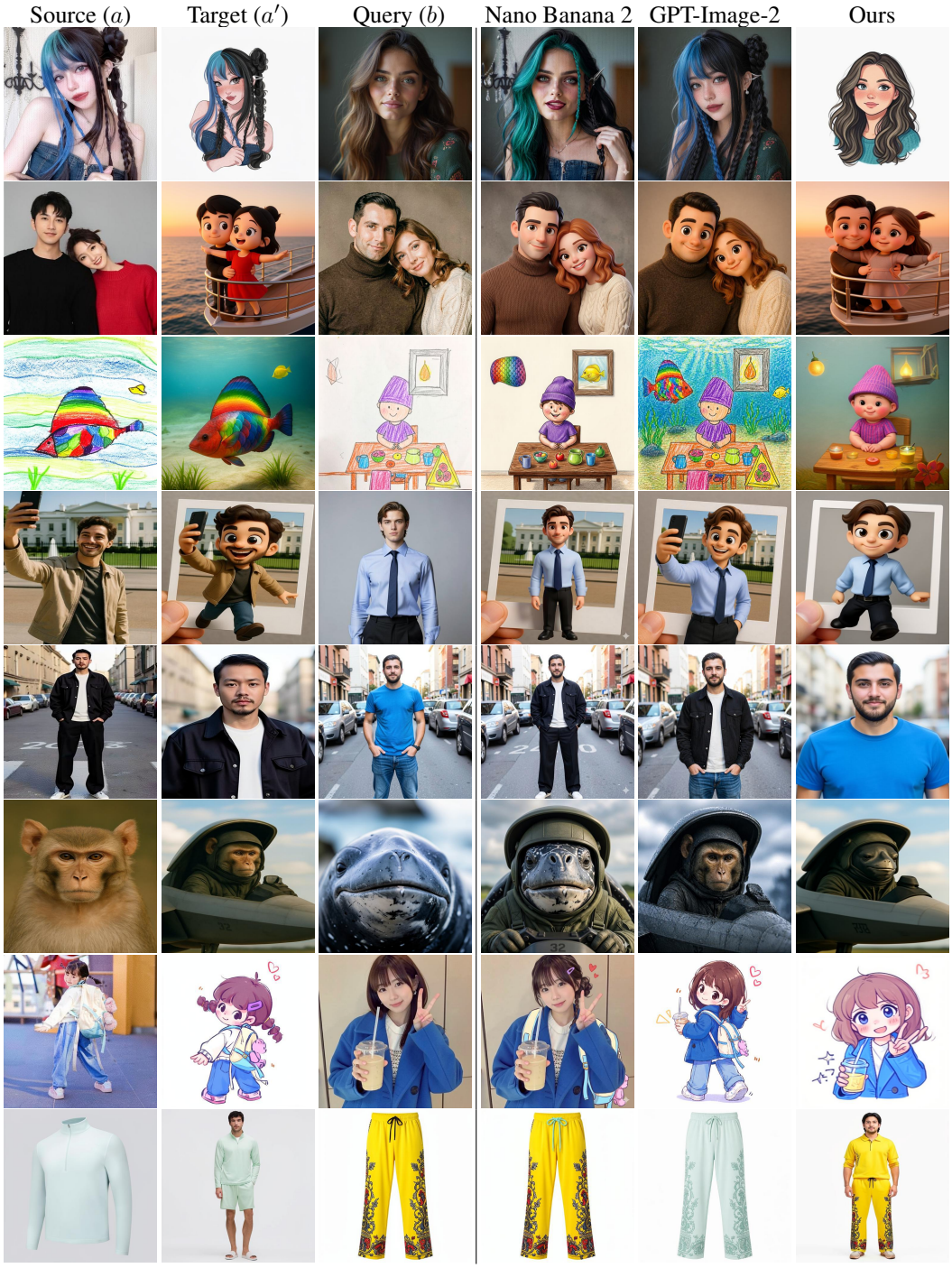


Figure 19: **Qualitative comparison with Nano Banana 2 [13] and GPT-Image-2 [38].** Both models frequently fail to capture the intended transformation (rows 1–3) and tend to leak appearance cues from the exemplar images into the output (rows 4–6).

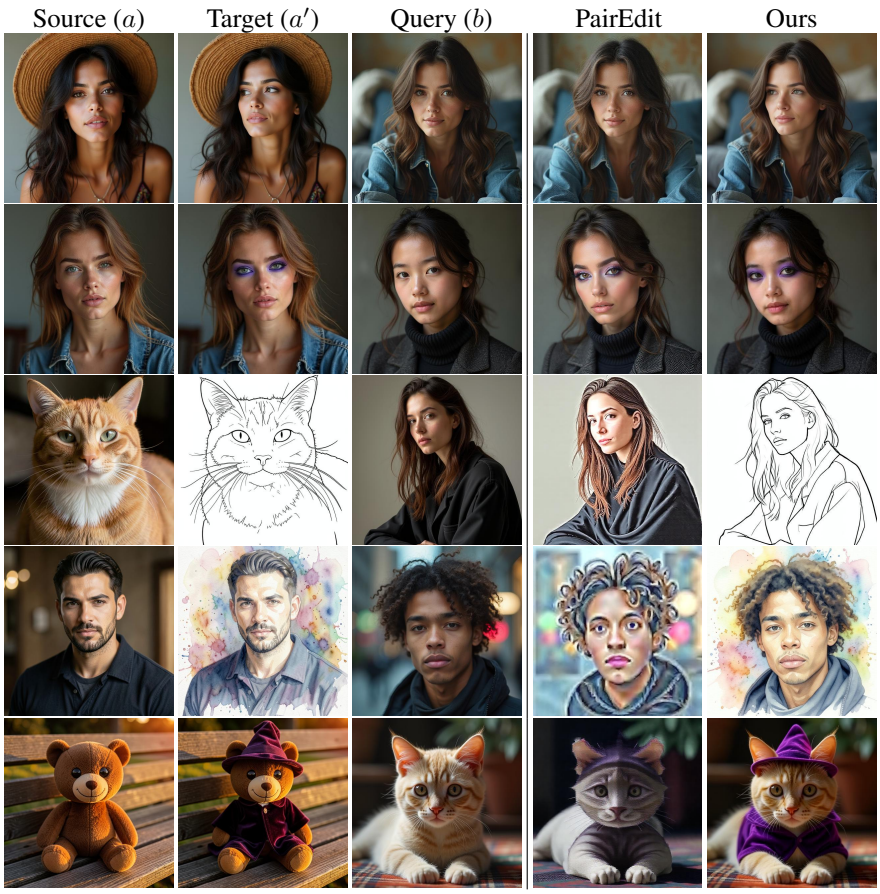


Figure 20: **Qualitative comparison with PairEdit [33].** PairEdit struggles to capture complex edits from the exemplar pair.



Figure 21: **Failure cases of Delta-Adapter.** Our method struggles with editing tasks that require precise text rendering. When the exemplar pair contains textual content, the model produces characters that are inconsistent with those in the exemplar.

Text Prompts

You are a strict visual evaluation assistant for reference-pair-guided image editing.

You will receive four images:
- A: reference source image
- B: reference target image
- C: source image to be edited
- D: candidate edited image

Your job:
1. Infer the intended edit from the actual visual transformation from $A \rightarrow B$.
2. Judge whether the same edit is correctly transferred from $C \rightarrow D$.
3. Score D on two dimensions:

GPT-A (Editing Accuracy): Whether the candidate correctly applies the intended edit or transformation shown by $A \rightarrow B$.

GPT-C (Content Consistency): Whether the candidate preserves the source image C everywhere outside the intended edit. Focus on non-target content preservation: identity, structure, geometry, layout, background, pose/camera, object count, and spatial relations should remain unchanged unless a change is clearly required by the intended edit shown in $A \rightarrow B$.

Core principle: The final judgment must rely only on the visual evidence in A, B, C, and D. Do not assume an edit category from filenames, directory names, or metadata.

General rules: Be strict. Do not reward unrelated changes. Score and explanation must be fully consistent. Judge transfer quality by comparing C with D under the transformation inferred from $A \rightarrow B$. Penalize GPT-C only for unwanted changes outside the target edit region.

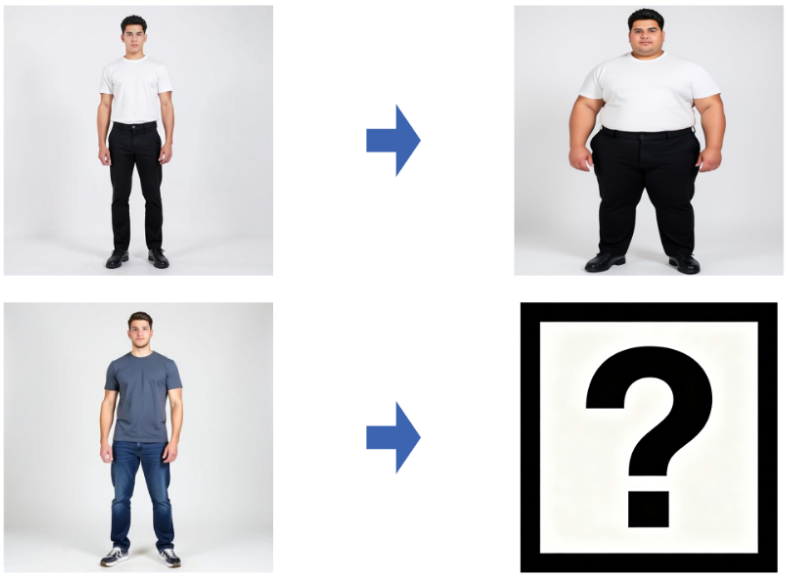
Output format. Return ONLY valid JSON with exactly this schema:

```
{
  "inferred_edit": "<one short sentence>",
  "D": {
    "scores": {
      "gpt_a": <integer 1-5>,
      "gpt_c": <integer 1-5>
    }
  }
}
```

Figure 22: System prompt used for GPT-based automated evaluation of editing accuracy (GPT-A) and content consistency (GPT-C).



Figure 23: **Visualization of recovered clean latents during four-step denoising.** Given a single exemplar pair (a, a') and a query image b, we visualize the decoded clean latent estimate $\hat{z}_0$ predicted at each denoising step of our four-step backbone. Even at the early denoising stages, the recovered $\hat{z}_0$ already forms coherent image structures and reflects the intended edit semantics, providing sufficiently reliable visual features for our dense semantic supervision.



Images in the first row: the editing effect from the left image to the right image is "changing the person to a fat body shape." Our goal is to apply the same editing effect to the image in the second row.
Please select the image from the options that has the better editing result. A good editing result means: 1) the editing effect is similar to the example in the first row, and 2) except for the edited region, the other regions should remain as consistent as possible with the original image.



B



A

Figure 24: **An example question from the user study.** Each question follows a two-alternative forced-choice format: participants view an exemplar pair (a, a'), a query image b, and two anonymized candidate edits produced by Delta-Adapter and a baseline, randomly assigned to positions A and B. Participants select the candidate that better reflects the transformation demonstrated by the exemplar pair while preserving the unedited regions of the query image.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: We have discussed the limitations of our method in Section 6.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide the theoretical derivation and necessary assumptions in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided the implementation details of our method in Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have submitted the code with sufficient instructions to reproduce the main experimental results. The code will be made publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided the experimental details in Sections 5.1 and further supplementary information in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: Error bars are not applicable to the experiments conducted in this paper. The qualitative comparison is the most important evaluation for this paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided the information on the computer resources in Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed the potential impacts of our work in the Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released assets are limited to code and evaluation scripts. We do not release high-risk pre-trained image generation or editing models, nor do we release scraped personal data.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have discussed the licenses of existing assets in the Appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code is accompanied by documentation including setup instructions and usage examples.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [Yes]

Justification: We have provided the full text of instructions given to participants and screenshots in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [No]

Justification: In our country, there is no equivalent organization for research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used as any important, original, or non-standard component of the core methodology. Their usage was limited to writing or editing support only.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.