LEARNING THE CHANGES: TRANSFERABLE VISUAL TRANSFORMATIONS FROM IMAGE-PAIR COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

A pair of images demonstrates a visual change precisely but binds it to the identity, layout, and texture of one scene. Existing systems typically learn to reuse the change from matched demonstration–query examples or represent it in a pretrained semantic space. We ask whether a reusable representation can instead emerge from reconstruction alone. A pair encoder compresses a source–target pair into a short edit code of a few continuous tokens, and a jointly trained diffusion editor reconstructs the target from it, with no text and no example of the edit acting on a second image. The learned codes apply the demonstrated change to new pairings of source images and transformations. Longer codes reconstruct the demonstrated pair more faithfully; shorter codes transfer more accurately and organize more clearly by transformation. The editor also applies two compatible changes at once from the concatenated codes of separate demonstrations, a format it never receives in training. This compositionality emerges from reconstruction as well, once the training targets themselves contain more than one change, as natural-image edits often do. The compressed representation thus retains the transformation itself, enough to carry it to new images and to combine it with another.

1 INTRODUCTION

A pair of images can demonstrate a change more precisely than a short verbal instruction. The pair may show how illumination shifts, how a material should look, or how a scene changes under a new viewpoint. It also contains information that should not be transferred, including the identity, layout, and texture of the example itself. Reusing the demonstration requires separating the transformation from the scene in which it appears.

Visual in-context learning typically provides direct supervision for this separation. Training examples contain a support pair, a different query image, and the desired query result. The model learns the same transfer operation that will later be requested at inference time (Wang et al., 2023b; Sun et al., 2023). Delta-Adapter offers a different solution. It defines the change as a displacement in frozen SigLIP2 features and encourages the generated edit to follow that displacement (Chen et al., 2026). These formulations show that image pairs can specify useful edits, and each supplies either a cross-image training signal or a semantic edit space chosen in advance. Whether either is necessary is an open question.

We investigate this question by restricting the information that can pass from a demonstration to an editor (Figure 1). A *pair encoder* maps one source and target pair to K continuous vectors, which we call an *edit code*. A separate diffusion editor receives the source and the code, and learns to reconstruct the same target. Transfer then tests what the bottleneck retains beyond the reconstruction task.

The bottleneck lets us test whether pair-specific detail and recurring visual changes compete for capacity, as one would expect. A wide code can afford details that help reconstruct one pair but say nothing about the transformation, while a narrow code has to spend its capacity on what the transformation itself requires. Wider codes should therefore reconstruct their own pairs more faithfully, and narrower codes should transfer more accurately and align across demonstrations of the same transformation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

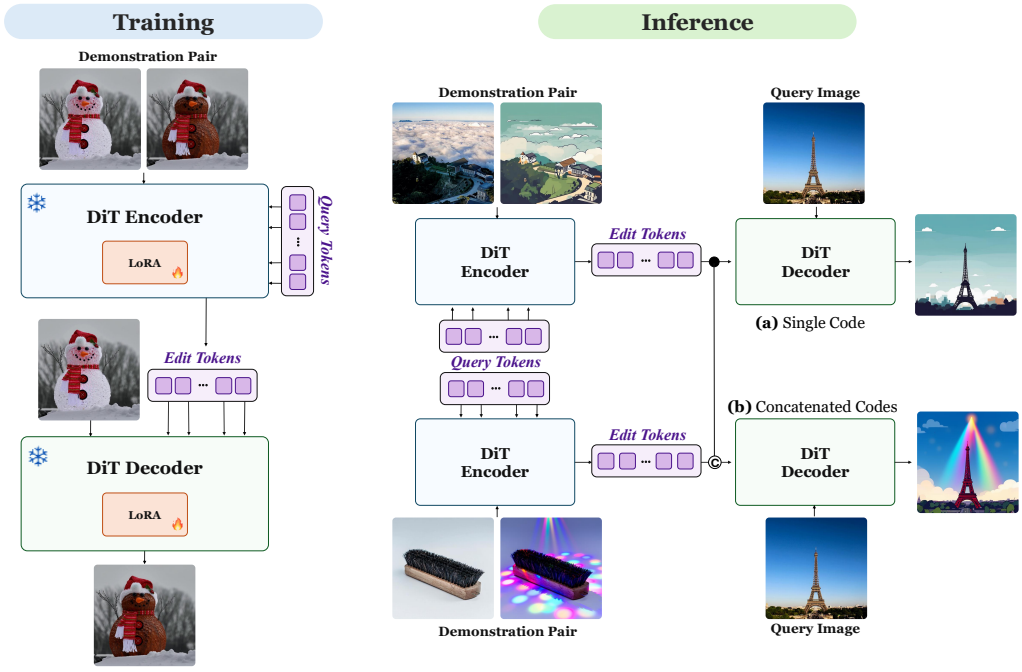

Figure 1: **Learning and reusing an edit code.** During training, the pair encoder compresses a demonstration into K edit tokens and the editor reconstructs its target. At inference, (a) one code transfers its edit to a new query, while (b) two codes inferred independently from different demonstrations are combined to control one output.

We test these predictions on 104,324 image pairs spanning diverse visual changes. Across a sweep of code lengths, increasing the code length improves reconstruction of the demonstrated pair but weakens transfer and the organization of the code space by transformation identity. Averaging codes from several demonstrations improves agreement with the desired edit. Concatenating two independently inferred codes can also produce both requested changes in one pass. We study the representation that this compression produces, its geometry and the composition that arises from it; we do not set out to build a general-purpose editor.

2 Related Work

Visual learning from examples. In-context learning conditions a model on examples without changing its parameters (Brown et al., 2020). Early visual formulations place support pairs and queries on a common canvas and train an inpainting or masked modeling objective (Bar et al., 2022; Wang et al., 2023a). Prompt Diffusion and ImageBrush train diffusion models on explicit support and query structures (Wang et al., 2023b; Sun et al., 2023). VisualCloze casts related generation tasks as image infilling over grids of examples (Li et al., 2025).

Two distinctions are especially relevant here: whether cross-image transfer is part of the training objective, and whether the edit is represented in a predefined semantic space. RelationAdapter, LoRWeB, and Edit Transfer are trained to infer a transformation from a demonstrated pair and apply it to a distinct query (Gong et al., 2025; Manor et al., 2026; Chen et al., 2025). All three retain textual conditioning. Delta-Adapter learns from a single pair without language, but derives its transformation from a frozen SigLIP2 encoder (Chen et al., 2026). Appendix F reports transfer on the Relation collection alongside these released pipelines.

A complementary line relies more heavily on the priors of an existing editor. Diffusion Image Analogies transfers a change specified by two images without requiring a user instruction (Šubrtová et al., 2023). Analogist adds structural attention and text generated by GPT-4V (Gu et al., 2024),

while Oorloff et al. (2025) modify attention computation in an otherwise fixed Stable Diffusion model. Visual Instruction Inversion instead maps image examples into the conditioning space of an existing editor (Nguyen et al., 2023).

Bottlenecked visual conditions. Information bottleneck methods seek representations that preserve information needed for a prediction while limiting information about the input (Tishby & Zaslavsky, 2015; Alemi et al., 2017). A closer architectural precedent is Paint by Example (Yang et al., 2023), which compresses a reference image to discourage direct copying into an inpainted region. That reference specifies the content to insert, whereas ours receives two images and must represent the change between them.

Latent changes and composition. Latent action models infer a compact transition variable from observations and use it to predict or control later states (Ye et al., 2025; Bruce et al., 2024). Work on distractors shows that such variables can absorb changes that are unrelated to the intended action (Nikulin et al., 2025). Composition has also been studied through visual analogies with structured tokens (Dedhia et al., 2023), score composition (Liu et al., 2022), and merging independently trained adapters (Shah et al., 2024). Here, two codes inferred from individual pairs are concatenated inside one editor, and no parameters are merged.

3 Learning an Edit Code through Reconstruction

3.1 Pair encoder and bottleneck

Let (x, y) denote a source image and its edited counterpart, with $y = T(x)$ for an unobserved transformation T. The pair encoder maps both images to

$$c = E_\phi(x, y) \in \mathbb{R}^{K \times d}. \tag{1}$$

The source and target are first mapped to the latent space of a frozen VAE. Their latent tokens are processed jointly by a diffusion transformer together with K learned query tokens. The hidden states at the query positions form the edit code c.

3.2 Reconstruction objective

A separate diffusion editor D_θ receives the clean source latent, a noised target latent y_t, timestep t, and the edit code c. Let v_t denote the target of the base model's rectified flow objective (Liu et al., 2023). We train the pair encoder and editor jointly with the standard rectified-flow loss, with the base model's timestep weighting $w(t)$, over single source–target pairs:

$$\mathcal{L}_{\text{rec}}(\phi, \theta) = \mathbb{E}_{(x,y),t}\Big[w(t)\, \|D_\theta(x, y_t, E_\phi(x, y), t) - v_t\|_2^2\Big]. \tag{2}$$

The editor receives x and y_t directly, so the clean target reaches it only through c. Both E_ϕ and D_θ are optimized.

3.3 Reusing the code

Given a support pair (x_s, y_s) and a new image x_q, we extract $c_s = E_\phi(x_s, y_s)$ once and sample

$$\hat{y}_q = \text{Sample}(D_\theta; x_q, c_s). \tag{3}$$

Multiple demonstrations of the same transformation are combined by averaging corresponding token slots, $\bar{c} = m^{-1} \sum_{i=1}^{m} E_\phi(x_i, y_i)$; the shared query bank fixes the slot correspondence. For two transformations A and B, we concatenate their codes:

$$\hat{y}_q^{A,B} = \text{Sample}(D_\theta; x_q, [c_A; c_B]). \tag{4}$$

3.4 Architecture and optimization

The pair encoder and editor are initialized from separate copies of the FLUX.2-klein-9B model (Black Forest Labs, 2026). The encoder processes source and target latent tokens at a fixed timestep of 0.5.

The editor concatenates clean source tokens with the noised target and predicts the target portion. In the principal configuration, the code is read from the encoder's text-conditioning stream and supplied to the editor's; Appendix E.5 varies both streams. Both transformers are adapted with LoRA (Hu et al., 2022), while the VAE and base weights remain frozen. The complete training and inference configuration appears in Appendix A.

4 A Dataset of Visual Changes

Testing whether an edit transfers beyond its demonstration requires examples of the same change in different scenes. We construct a dataset in which visual edits recur across different source images drawn from the video collection of VAP-Data (Bian et al., 2026). The edits span styling; lighting, color, and atmosphere; viewpoint and geometry; scene edits; and subject edits. The resulting collection contains 104,324 accepted source and target pairs; only the VAP-Data frames are used.

Targets are produced through a mixture of direct edit instructions, specialized LoRAs, and deterministic image operators. A vision and language model adapts instructions to individual images and filters unsuccessful generations. Figure 2 summarizes the scale and visual range of the collection. Appendix C describes source selection, target generation, and splits; Appendix D shows more examples from each group.



Figure 2: **Dataset scale and visual diversity.** The dataset contains 104,324 source and target pairs. The chart on the left shows their distribution across five broad edit groups. The first gallery tile is the shared source; the remaining images illustrate edits from across the dataset, with colored rules indicating their group.

5 Experimental Questions and Protocol

A code that reconstructs its target could carry the transformation, or it could carry the appearance and layout of the demonstration. We ask three questions that distinguish these possibilities.

Does the code transfer, and do multiple supports agree? For the support count experiment, a fixed query set is edited using one, two, or four demonstrations of the requested transformation. Codes from multiple demonstrations are averaged by token position. Predictions are compared with the target after a VAE round trip using pixel metrics and features from CLIP (Radford et al., 2021), SigLIP2 (Tschannen et al., 2025), and DINOv2 (Oquab et al., 2024). Directional agreement compares the feature displacement from query to prediction with the displacement from query to target. A Qwen3.5-9B judge (Qwen Team, 2026) sees the support pair, query, and prediction, but not the paired query target. It scores transformation transfer, source preservation, absence of unintended changes, realism, and overall quality from one to five. Prior work finds that multimodal judges can be useful for conditional image evaluation but remain less reliable for image editing and when using open models (Ku et al., 2024).

Does stronger compression produce a more selective code space? We mean-pool the K vectors for every validation pair and measure whether nearby codes share a transformation identity. Label-

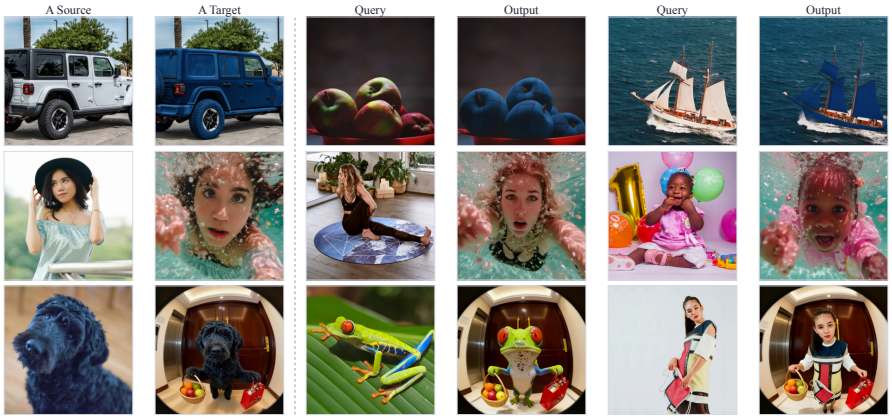

Figure 3: **Transferring an edit code to a query image.** The pair encoder observes the demonstration pair on the left, and its code conditions the editor on two different query images. Each row reuses the same demonstration for both query–output pairs.

aware measures include cosine nearest-neighbor accuracy and leave-one-out nearest-centroid accuracy. We also cluster the codes without labels and measure agreement with transformation identities using adjusted Rand index, alongside the full set of clustering measures in Appendix E.3.

Can independently inferred codes be combined? We test whether codes inferred independently from two demonstrations can jointly control a single output. For transformations A and B, we condition on $[c_A; c_B]$ and $[c_B; c_A]$, sampling task pairs from predefined cross-family groups and pooling both concatenation orders. Image metrics compare each single-pass output with the corresponding two-pass sequential result as a proxy reference. Because some edit pairs combine into a single joint result without a unique order, we also use reference-free VLM and human evaluations. Both evaluations observe the demonstrations, query, and single-pass output and assess whether each edit and their combination are present.

6 Results

6.1 Reconstruction alone produces a transferable edit code

Figure 3 shows transfer from one demonstration to two query images. The inferred code applies a terry-cloth material to a new object, places a new scene under water, and transfers a fisheye viewpoint. In the third row the demonstration puts the dog in a fisheye elevator with a red bag, and both outputs receive the viewpoint, the setting, and the bag; the code carries the whole demonstrated change. Additional examples appear in Appendix I.1.

Averaging codes from four independent demonstrations improves every metric that compares the prediction with the paired target, including perceptual distance and directional alignment (Table 1), while the reference-free semantic judgment remains essentially unchanged. Averaging improves agreement with the particular target realization more than it improves the judged quality of the edit. Complete image and semantic metrics appear in Appendix E.2.

Table 1: Averaging demonstrations for $K = 2$ on 352 fixed queries. Bold marks the best value and underline the second best.

	Target distance ↓		Edit direction ↑	VLM ↑
Supports	CLIP	SigLIP2	CLIP	Overall / 5
1	0.112	0.093	0.766	<u>4.37</u>
2	<u>0.096</u>	<u>0.081</u>	<u>0.800</u>	**4.41**
4	**0.092**	**0.076**	**0.810**	4.34

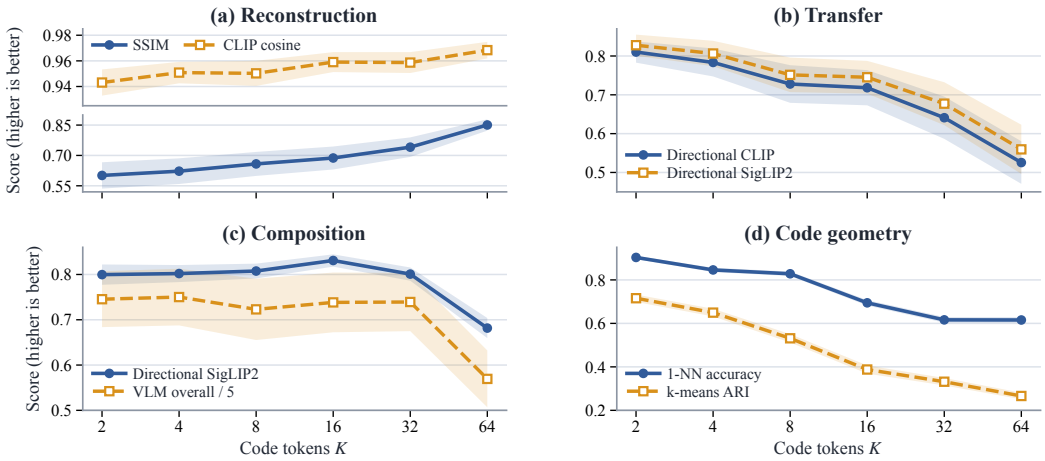

Figure 4: **Wider codes reconstruct better but reuse worse.** Each code length K is a separately trained checkpoint. (a) Reconstruction of the demonstrated target from its own code. (b) Transfer to a disjoint query from the mean of four support codes. (c) One-pass composition of two concatenated codes, pooled over both orders. (d) Organization of the held-out code space by transformation identity. Shading marks 95% bootstrap intervals (Appendix E).

6.2 Shorter codes transfer more accurately

Across separately trained configurations, same-pair reconstruction improves steadily with code length (Figure 4); SSIM rises from 0.601 at $K = 2$ to 0.850 at $K = 64$.

Transfer becomes less accurate as code length increases. With four supports, directional CLIP falls from 0.810 to 0.526, and transformation identity becomes less distinct in code space. Longer codes appear to retain source and target details that help reconstruct the original pair but are poorly shared across images. The tradeoff is clearest at the endpoints. Every one of the 44 evaluated transformation identities transfers better at $K = 2$ than at $K = 64$, but adjacent code lengths are not separated as cleanly. At $K = 4$, fewer than half of the identities show all three signs (higher SSIM and lower directional CLIP and SigLIP2). Composition remains broadly stable through the intermediate code lengths and then declines at the widest setting. Per-identity contrasts and complete metrics for all four evaluations appear in Appendix E.

Transformation identity is recoverable from the code space. At $K = 2$, a code's nearest neighbor shares its transformation identity in 90.3% of validation cases. Leave-one-out nearest-centroid accuracy is 94.6%. These measurements use the original code space and labels withheld from training. Their decline at larger K quantifies the loss of transformation separation.

The t-SNE visualization (van der Maaten & Hinton, 2008) in Figure 5 shows the same five semantic groups used to summarize the dataset in Figure 2. Within the region associated with lighting, color, and atmosphere, individual transformations form smaller neighborhoods.

Similar transfer also appears when the code is routed through FLUX's image stream at both stages (Appendix E.5). This alternative route weakens composition, so transfer and joint control respond differently to how the editor receives the code.

6.3 A controlled benchmark with exact composition targets

Natural-image edits often have several plausible joint outputs, making exact composition difficult to measure. We use colored-object scenes in which each rule recolors objects selected by shape, size, or position. Separate demonstrations specify rules A and B, and their independently inferred codes are applied to a new query (Figure 6). Symbolic rule descriptions generate the targets and metrics. We first consider compatible pairs, for which applying the rules in either order gives the same target,

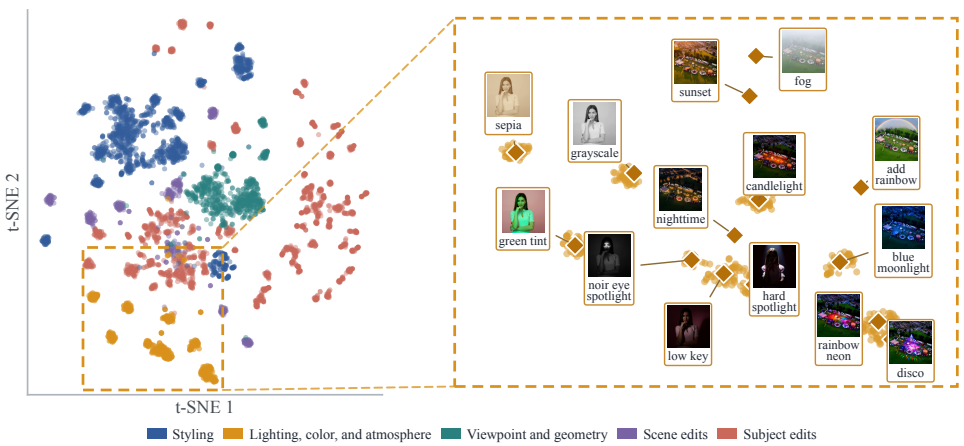

Figure 5: **Organization by transformation identity.** Left: cosine t-SNE of the mean-pooled held-out $K = 4$ edit codes, colored by the five groups from Figure 2. Right: the dashed coordinate window is enlarged and shows the transformation centroids from the lighting, color, and atmosphere group. Each diamond is paired with an actual dataset target generated from one of two fixed source images.

$T_B(T_A(x)) = T_A(T_B(x))$. This includes rules that act on disjoint objects and overlapping rules that assign the same color.

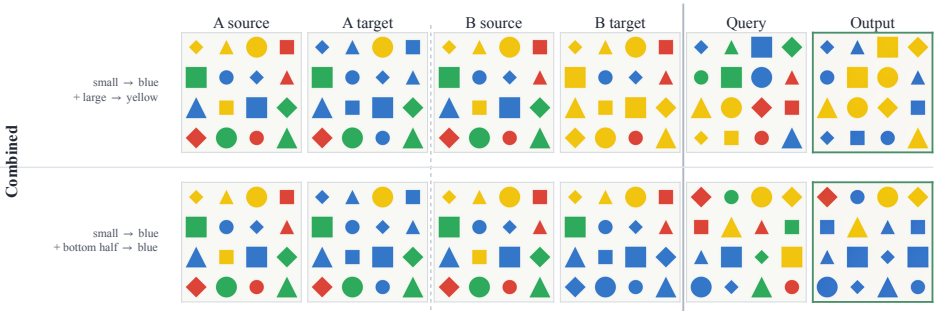

Figure 6: **Exact one-pass composition in controlled scenes.** Each row specifies rules A and B through separate visual demonstrations. Their codes are inferred independently, concatenated, and applied once to the query. The first example uses disjoint rules; the second uses overlapping rules that assign the same color. Both outputs satisfy the two indicated rules exactly.

We train two checkpoints that differ in whether the editor has ever rendered a joint target. Many natural-image targets combine several changes, as the fisheye elevator with the red bag in Figure 3 shows, so the natural-image editor learns to render more than one change in a single target even though each pair is labeled as one edit. In the controlled scenes every atomic rule makes exactly one change, which lets us withhold that experience. Atomic is trained only on single-rule targets. Combined also reconstructs joint targets from one code and trains for more optimizer steps (Appendix H); neither checkpoint ever receives concatenated codes during training, and the component rules appear individually in training while their evaluated pairings are held out.

When two independently inferred codes are concatenated at test time, Combined reaches 90.8% exact-scene success on compatible pairs, compared with 10.3% for Atomic (Table 2). Addition and averaging, which merge the codes back to the original length, remain below 23% for both checkpoints. Atomic can still apply the edits in two separate passes, but having fit only single-rule targets it does not produce their exact joint target in one pass. Once the editor has learned that it can apply several rules to one target, composition follows for pairings it has never seen, as long as the two code blocks remain separate.

Because the two recipes differ in both joint-target exposure and training duration, this contrast does not isolate the effect of either. Appendix H breaks down the compatible results by pair type and examines order-sensitive rules.

Table 2: Exact-scene success (%) for one-pass composition over 128 held-out compatible transformation pairs, with four queries per pair and both code orders. Each pair has the same target in both orders.

	Exact scene (%) ↑	
One-pass inference	Atomic	Combined
Add: $c_A + c_B$	<u>8.8</u>	21.3
Average: $(c_A + c_B)/2$	6.6	<u>22.7</u>
Concatenate: $[c_A; c_B]$	**10.3**	**90.8**

6.4 Concatenated codes produce joint control in natural images

We also test concatenation in natural images, where there is no unique exact target but the requested changes remain visually recognizable (Figure 7). The natural-image editor is trained only on single-edit pairs, but many of those targets already contain several changes, as the previous section notes, so we expect concatenated codes to compose here as well. One code pair renders snowy weather in the demonstrated vector style; another adds a crown while rendering the query in the demonstrated cartoon style. On a two-token image-to-text checkpoint trained further at 480×480 pixels, human raters scored single-pass outputs from one to five (Appendix E.6). They saw both edits applied (per-edit application 3.86) and the query kept intact (source preservation 4.35), and they were less sure that the two combined as intended (composition success 3.77, 95% CI 3.53–4.00). Without an exact natural-image target, the study measures whether joint control is recognizable; the exact measurement comes from the controlled benchmark. Appendices I.2 and E.4 give more examples and the automatic evaluation.

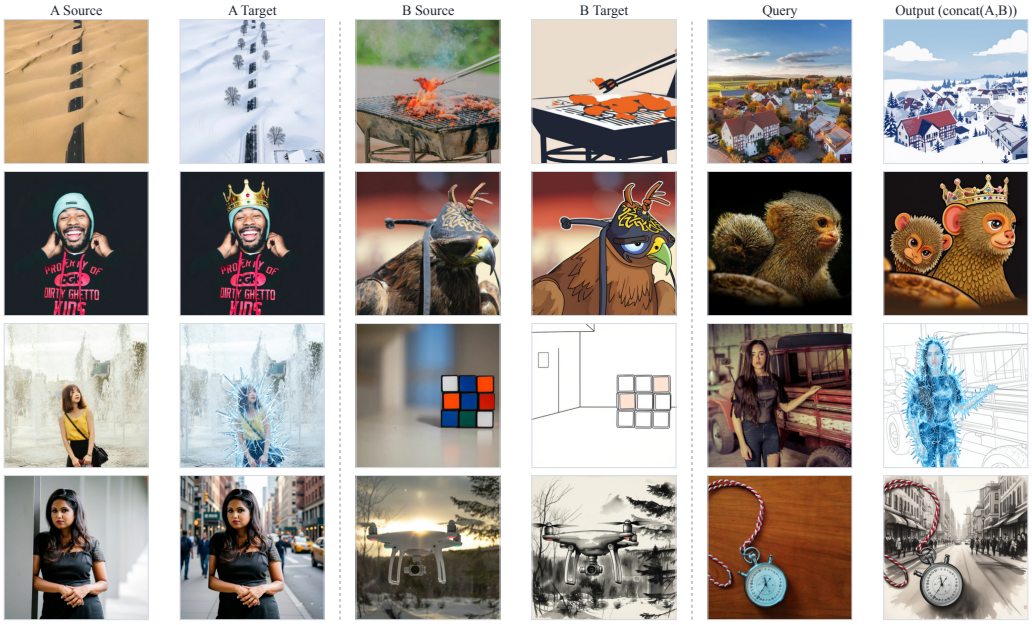

Figure 7: **Joint control by concatenated edit codes.** Each row begins with two visual demonstrations. Their independently inferred codes are concatenated and applied once to the held-out query; the final column is the resulting single-pass output.

7 DISCUSSION

Because the editor already receives the source image, the code only has to convey enough of the change to distinguish the target from other possible edits. A short code conveys enough of it to move the change onto other images, so the transformation survives the compression while most of the demonstration's appearance does not. Wider codes reconstruct their own pair better and transfer worse; the added capacity goes to details of the particular realization.

The code space supports this reading. Codes of the same transformation are neighbors across scenes, and the editor's agreement with the requested target improves when it consumes the mean of several independently inferred codes, so separate demonstrations of one change produce codes that are compatible enough to average.

Averaging demonstrations of one transformation and combining different transformations also place different demands on the editor. The former retains what the supports share; the latter must preserve two distinct changes. Concatenation hands the editor both edit codes intact, whereas averaging merges two edits into one sequence that the editor must then untangle. The controlled benchmark shows the resulting advantage of concatenation over addition and averaging. Two intact code blocks do not suffice for Atomic, however, which applies the same edits in separate passes but not in one. Joint control thus depends on how the editor interprets the codes, as well as on what the encoder retains. Routing the code through different conditioning streams shows the same dependence, since routes with similar transfer differ in composition quality (Appendix E.5).

The idealized analysis in Appendix G makes the capacity tradeoff explicit. Once recurring variation is retained, encoding a support-specific detail lowers reconstruction error while raising error on an independent query. The analysis assumes a fixed split between recurring and private variation, so it explains the direction of the tradeoff but not the token count at which transfer peaks.

8 LIMITATIONS

The code does not always isolate the transformation cleanly: predictions can retain appearance from the demonstration. Moreover, targets come from several editors, LoRAs, prompted procedures, and deterministic operators. This breadth leaves open how much of the code geometry reflects the intended change rather than the route that produced it. Generating the same transformations through independent routes would help separate these effects.

The natural-image dataset supplies targets for individual edits, so it cannot establish exact composition accuracy. Some edit pairs also impose incompatible changes on the same content, which a single output cannot satisfy simultaneously. The controlled study supplies exact targets for simple recoloring rules. Extending exact evaluation to richer natural-image transformations remains open.

9 CONCLUSION

Reusing a demonstrated edit has typically relied on matched demonstration and query examples or on a semantic space fixed in advance. We find that a compressed token representation of the pair, learned only to reconstruct the demonstrated target, already carries the transformation. It moves a material, a setting, or a viewpoint onto a new image, and codes inferred from separate demonstrations of the same change agree closely enough to be averaged. Adding capacity improves reconstruction of the demonstrated pair but weakens the reuse of the change, so what survives the compression is the part of the demonstration that generalizes.

These codes also combine with one another. An editor that has learned to render targets with several changes applies two demonstrated edits at once when their codes are concatenated, on pairings it has never seen. Reconstruction through a narrow bottleneck yields a representation of the change that can be transferred, averaged, and composed.

AI USE STATEMENT

Generative AI tools were used to help draft and polish the manuscript, retrieve and organize related literature, prepare figures and tables, and implement supporting code. The research questions, method, experimental design, interpretation of the results, and scientific conclusions were developed by the authors. The authors reviewed all AI-assisted text and artifacts, tested AI-assisted code, verified reported results against saved experimental outputs, and checked citations against the source papers. The authors take responsibility for the final content. Synthetic target images were generated using the models and procedures described in Appendix C.

ETHICS STATEMENT

The source images are frames from the VAP-Data video collection (Bian et al., 2026), and every target is generated by the editing models, adapters, and operators listed in Appendix C; no new images of people were collected. The method edits images from a visual demonstration and carries the risks of any image editor, including the manipulation of depictions of real people and the imitation of artistic styles. The human study is described in Appendix E.6: raters were volunteers, only their ratings were stored, and we assessed that the study did not require ethical review.

REPRODUCIBILITY STATEMENT

Appendix A describes the model and training setup; Appendix B specifies the evaluation protocol and Appendix J the metric definitions; and Appendix C documents data construction and filtering. Full numerical results, the controlled composition study, and additional qualitative examples appear in Appendices E, H, I.1, and I.2.

REFERENCES

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2017.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989. doi: 10.1016/0893-6080(89)90014-2.

Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei A. Efros. Visual prompting via image inpainting. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Yuxuan Bian, Xin Chen, Zenan Li, Tiancheng Zhi, Shen Sang, Linjie Luo, and Qiang Xu. Video-As-Prompt: Unified semantic control for video generation. In *International Conference on Learning Representations*, 2026.

Black Forest Labs. FLUX.2 [klein]: Towards interactive visual intelligence. Black Forest Labs blog, 2026. URL `https://bfl.ai/blog/flux2-klein-towards-interactive-visual-intelligence`. Accessed 2026-09-23.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623, 2024.

Jiacheng Chen, Songze Li, Han Fu, Baoquan Zhao, Wei Liu, Yanyan Liang, Li Qing, and Xudong Mao. Delta-Adapter: Scalable exemplar-based image editing with single-pair supervision. *arXiv preprint arXiv:2605.07940*, 2026.

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.

Bhishma Dedhia, Michael Chang, Jake C. Snell, Thomas L. Griffiths, and Niraj K. Jha. Im-Promptu: In-context composition from image prompts. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Xiangyu Fan, Zesong Qiu, Fanzhou Wang, Peng Gao, and Lei Yang. Qwen-Image-Lightning. GitHub repository, 2025. URL `https://github.com/ModelTC/LightX2V-Qwen-Image-Lightning`. Accessed 2026-09-23.

Yan Gong, Yiren Song, Yicheng Li, Chenglin Li, and Yin Zhang. RelationAdapter: Learning and transferring visual relation with diffusion transformers. In *Advances in Neural Information Processing Systems*, volume 38, pp. 97367–97393, 2025. doi: 10.52202/085713-3255.

Zheng Gu, Shiyuan Yang, Jing Liao, Jing Huo, and Yang Gao. Analogist: Out-of-the-box visual in-context learning with image diffusion model. *ACM Transactions on Graphics*, 43(4):1–15, 2024. doi: 10.1145/3658136.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Max Ku, Dongfu Jiang, Cong Wei, Xiang Yue, and Wenhu Chen. VIEScore: Towards explainable metrics for conditional image synthesis evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12268–12290, 2024. doi: 10.18653/v1/2024.acl-long.663.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 18969–18979, 2025.

Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pp. 423–439, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Hila Manor, Rinon Gal, Haggai Maron, Tomer Michaeli, and Gal Chechik. Spanning the visual analogy space with a weight basis of LoRAs. In *European Conference on Computer Vision*, 2026.

Thao Nguyen, Yuheng Li, Utkarsh Ojha, and Yong Jae Lee. Visual instruction inversion: Image editing via image prompting. In *Advances in Neural Information Processing Systems*, 2023.

Alexander Nikulin, Ilya Zisman, Denis Tarasov, Nikita Lyubaykin, Andrei Polubarov, Igor Kiselev, and Vladislav Kurenkov. Latent action learning requires supervision in the presence of distractors. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 46427–46447, 2025.

Trevine Oorloff, Vishwanath Sindagi, Wele Gedara Chaminda Bandara, Ali Shafahi, Amin Ghiasi, Charan Prakash, and Reza Ardekani. Stable Diffusion models are secretly good at visual in-context learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23604–23613, 2025.

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. *Transactions on Machine Learning Research*, 2024.

Qwen Team. Qwen3.5-9B. Hugging Face model card, 2026. URL `https://huggingface.co/Qwen/Qwen3.5-9B`. Accessed 2026-09-23.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Viraj Shah, Nataniel Ruiz, Forrester Cole, Erika Lu, Svetlana Lazebnik, Yuanzhen Li, and Varun Jampani. ZipLoRA: Any subject in any style by effectively merging LoRAs. In *European Conference on Computer Vision*, 2024.

Adéla Šubrtová, Michal Lukáč, Jan Čech, David Futschik, Eli Shechtman, and Daniel Sýkora. Diffusion image analogies. In *ACM SIGGRAPH 2023 Conference Proceedings*, pp. 1–10, 2023. doi: 10.1145/3588432.3591558.

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 *Advances in Neural Information Processing Systems*, 2023.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop*, pp. 1–5, 2015. doi: 10.1109/ITW.2015.7133169.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6830–6839, 2023a.

Zhendong Wang, Yifan Jiang, Yadong Lu, Yelong Shen, Pengcheng He, Weizhu Chen, Zhangyang Wang, and Mingyuan Zhou. In-context learning unlocked for diffusion models. In *Advances in Neural Information Processing Systems*, 2023b.

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

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-Image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18381–18391, 2023.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations*, 2025.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. doi: 10.1109/CVPR.2018.00068.

A Implementation Details

The pair encoder and editor are independent FLUX.2-klein-9B transformers. The frozen VAE maps both images to latent tokens. The pair encoder processes these tokens together with K learned queries, and the hidden states at the query positions form the code. The editor concatenates a clean source latent with a noised target latent and predicts only the target portion. The default route reads and writes the code through the text-conditioning stream.

The base transformer weights and VAE remain frozen. Rank 128 LoRA modules are added to both transformers, covering the linear layers of repeated transformer blocks whose weight dimension is at least 512. The learned query bank in the pair encoder is optimized directly. Table 3 gives the training hyperparameters, and Table 4 specifies the sampling and evaluation protocol. The edit code is dropped for 20% of training pairs, which provides the unconditional branch for classifier-free guidance at sampling time.

Table 3: Training configuration for the code length and routing ablations. Values are shared by the pair encoder and editor unless stated otherwise.

Parameter	**Value**
Architecture	
Backbone	Two independent FLUX.2-klein-9B transformers
Image autoencoder	Shared frozen FLUX.2 VAE
Frozen parameters	Base transformer weights and VAE
Code configuration	
Code length K	$\{2, 4, 8, 16, 32, 64\}$
Code space	Inner transformer representation
Query initialization	Normal distribution, standard deviation 0.02
Pair encoder condition	Fixed timestep 0.5; embedded guidance 0
Code dropout probability	0.2
LoRA configuration	
Rank	128
Adapted modules	Pair encoder DiT and editor DiT
Target modules	Repeated-block linear layers with minimum weight dimension 512
Additional trainable parameters	Pair encoder query bank
Training configuration	
Image resolution	256×256
Per-process batch size	6
Training processes	4
Gradient accumulation steps	1
Global batch size	24
Data workers	8
Optimizer steps	45,000
Optimizer configuration	
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	0.01
Learning-rate schedule	Constant
Noise configuration	
High-noise warmup	2,500 steps
Timestep index sampler	Log-normal, mean -1.5, standard deviation 1.0

B Evaluation Configuration

Table 4: Evaluation protocol shared by the code length sweep. The routing sweep uses the same checkpoint step and metrics but a separate evaluation sample (Appendix E.5). Sample counts include examples satisfying the grouping requirements of each experiment.

Evaluation	Configuration
Sampling configuration	
Checkpoint and images	Step 45,000; 256×256 pixels; sample seed 0
Sampler	30 steps; denoiser guidance 3.0; classifier-free guidance scale 3.0
Evaluation tasks	
Same-pair reconstruction	Code inferred from one validation pair and applied back to its source; the same 44 pairs are used for every K
Transfer	44 transformation identities; one, two, or four shared demonstrations; eight queries per identity; 352 queries per row
Code geometry	All 5,268 validation pairs and all 171 transformations; mean-pooled code space
Composition	50 task pairs drawn from 16 predefined family pairings; one support per task; queries drawn from both source tasks; 380 outputs per order
Reference configuration	
Image reference	Target after VAE encoding and decoding for transfer; sequential application for composition
Evaluation models	
Feature models	CLIP ViT-L/14, SigLIP2 base patch 16 at 224 pixels, and DINOv2 large
VLM judge	Qwen3.5-9B; longest image side 512 pixels; structured scores from one to five

C Dataset Construction

Source imagery. The 1,217 source images are selected from the video portion of VAP-Data (Bian et al., 2026). They span people, animals, objects, landscapes, and scenes with multiple subjects, which we group into six broad source categories. Only the selected frames are used, as source images.

Display groups. For Figure 2, we assign the 171 transformations to five broader groups, splitting a construction family when it mixes operation types. *Styling* covers artistic renderings, character styles, cosplay, and animal retheming. *Lighting, color, and atmosphere* includes relighting, grayscale and sepia operators, changes in time of day or atmosphere, and a noir spotlight treatment. *Viewpoint and geometry* collects camera changes, rotations and flips, and special perspective or scale edits. *Scene edits* comprise changes to backgrounds and other scene elements, including removal. *Subject edits* alter the focal subject or its relation to another object; these include local recoloring, materials and physical properties, clothing and accessories, animal replacement, merchandise rendering, held objects, and riding. These groups are used only to summarize the dataset visually. Training and evaluation retain all 171 transformation identities and the 20 original construction families.

Target generation and filtering. We define 171 transformations. For 69 of them, `Qwen/Qwen3-VL-8B-Instruct-FP8` (Bai et al., 2025) converts a transformation template into an instruction and target description tailored to the source image. Another 95 use fixed transformation-specific prompts, and the remaining seven apply deterministic geometric or color operations. Generative targets are produced by Qwen-Image-Edit (Wu et al., 2025) with a four-step Lightning model (Fan et al., 2025) and zero guidance, optionally augmented with a transformation-specific LoRA. The same vision and language model then checks the source, target, and intended change. A pair is retained only if the requested edit is present, unrelated content is preserved, and no severe visual or geometric failure is detected. Table 5 summarizes the target-generation methods. The final collection contains 104,324 pairs, with 99,056 used for training and 5,268 for validation.

Table 5: Target-generation methods. "QIE" abbreviates Qwen-Image-Edit; each learned method uses the corresponding four-step LightX2V Lightning model. Counts denote transformations rather than generated pairs.

Route	#	Editor	Add-on checkpoint
Prompted edits	43	QIE-2511	none
Art styles	20	QIE-2511	`svjack/OmniConsistency`
Camera viewpoints	6	QIE-2511	`fal/Multiple-Angles`
Ride an animal	38	QIE-2509	`valiantcat/rideanimals`
Material transfer	17	QIE-2509	`oumoumad/Material-transfer`
Character cosplay	10	QIE-2509	`valiantcat/Cosplay`
Relighting	7	QIE-2509	`dx8152/Relight`
Animal retheming	6	QIE-2509	`valiantcat/Zootopia`
Merchandise rendering	5	QIE-2509	`svjack/Goods`
Object removal	3	QIE-2511	`Ghazaleruv431/Object-Remover`
Outfit extraction	1	QIE-2511	`prithivMLmods/Extract-Outfit`
Phone-screen viewpoint	1	QIE-original	`valiantcat/ARStandonScreen`
Fisheye viewpoint	1	QIE-original	`valiantcat/fisheye`
Giant figurine	1	QIE-original	`starsfriday/Giant-Figurine`
Underwater scene	1	QIE-original	`QwenEdit_InWater` adapter
Noir spotlight	1	QIE-2511	`prithivMLmods/Noir-Spotlight`
Comic-book panel	1	QIE-2511	`prithivMLmods/Noir-Comic`
3D animation	1	QIE-2511	`prithivMLmods/PI3`
2D avatar	1	QIE-original	`valiantcat/2dCartoonAvatar`
Exact image operators	7	deterministic	grayscale, sepia, horizontal/vertical flip, and 90° CW/CCW or 180° rotation

Training uses only the resulting image pairs, without identifying how each target was produced. The same pair encoder and editor are used across all transformations and generation methods.

D Transformation Atlas

Figure 8 shows a source and target example for each of the 20 transformation families.

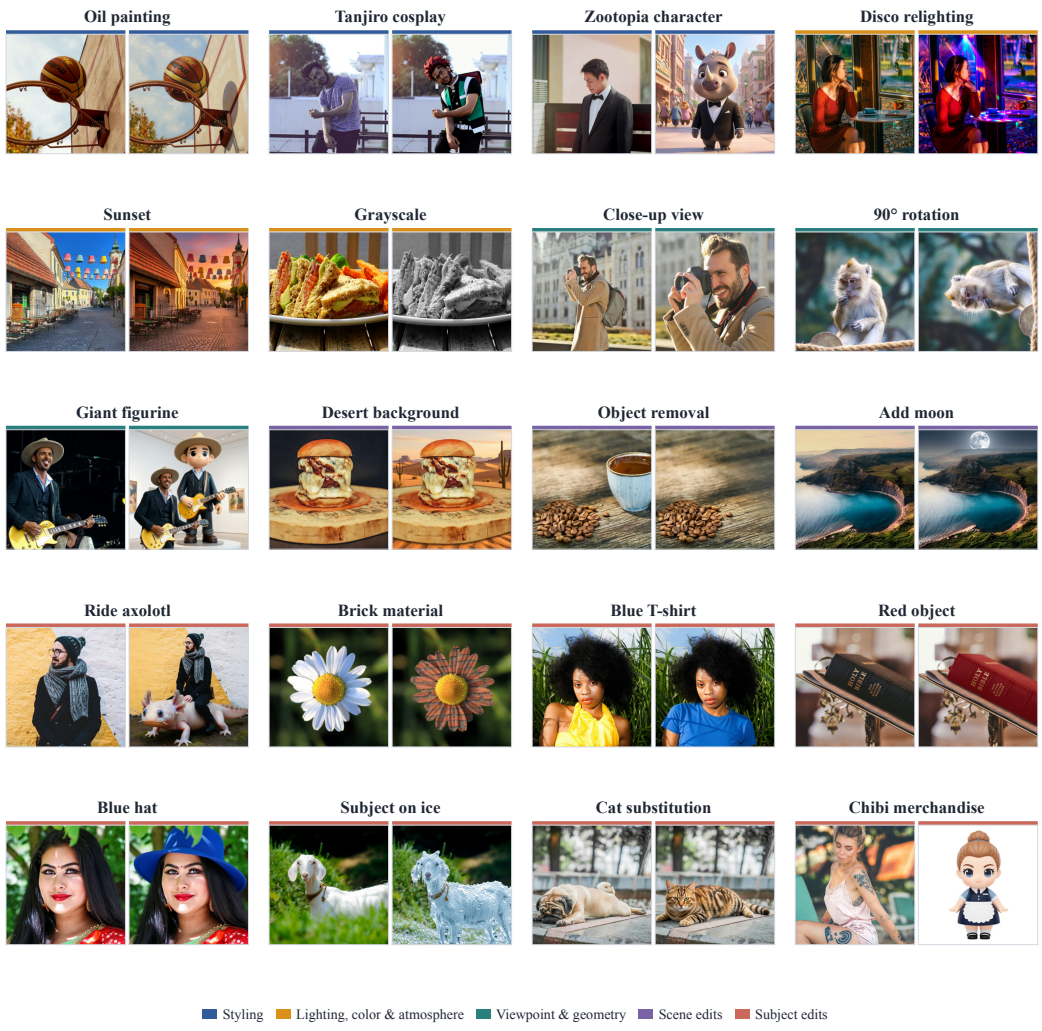

Figure 8: **Visual taxonomy of the transformation dataset.** Each cell shows a source and edited target from one family. Colored rules use the five display groups introduced in Figure 2.

E Additional Quantitative Results

In the following ablations, bold denotes the best value within each comparable column; ties at the reported precision are also bold.

The confidence intervals in Figure 4 use bootstrap standard errors from 10,000 replicates. Reconstruction and transfer resample transformation identities, while composition resamples unordered transformation pairs. For code geometry, we resample held-out codes: 1-NN accuracy is recomputed from the binary same-identity neighbor outcomes, and adjusted Rand index from the saved identity–cluster assignments.

E.1 Reconstruction across bottleneck sizes

Table 6 reports how well each model reconstructs the pair from which its code was inferred. The same 44 held-out pairs are used at every code length, and predictions are compared with targets after the frozen VAE round trip. Reconstruction improves as the bottleneck widens. This trend is the reverse of cross-image transfer and code geometry; the wider checkpoints keep their editing capacity even as their codes become less reusable.

Table 6: Same-pair reconstruction across code lengths. Each code is applied back to its own source. Inc. denotes Inception feature distance. Best values are bold.

K	n	MSE↓	MAE↓	PSNR↑	SSIM↑	Inc.↓	CLIP↓	SigLIP2↓	DINOv2↓
2	44	0.020	0.078	18.37	0.601	7.69	0.057	0.037	0.077
4	44	0.019	0.072	19.18	0.622	7.27	0.049	0.033	0.068
8	44	0.015	0.064	20.03	0.658	7.05	0.050	0.032	0.065
16	44	0.013	0.061	20.65	0.687	6.77	0.041	0.034	0.058
32	44	0.009	0.050	22.01	0.741	6.14	0.041	0.025	0.040
64	44	**0.004**	**0.036**	**24.77**	**0.850**	**5.47**	**0.032**	**0.020**	**0.034**

E.2 Support aggregation and transfer

Four-support averaging improves every metric that uses the paired target. The reference-free overall judgment remains essentially flat, however, with no monotonic benefit from additional supports (Tables 7 and 8).

Table 7: Image metrics for support averaging at $K = 2$. Each row evaluates the same 352 queries against targets after a VAE round trip. Inc. is Euclidean distance in Inception features.

Supports	n	MSE↓	MAE↓	PSNR↑	SSIM↑	Inc.↓	CLIP↓	SigLIP2↓	DINOv2↓
1	352	0.059	0.162	13.69	0.439	10.72	0.112	0.093	0.192
2	352	0.050	0.146	14.41	0.457	10.06	0.096	0.081	0.162
4	352	**0.046**	**0.138**	**14.84**	**0.469**	**9.96**	**0.092**	**0.076**	**0.154**

Table 8: Directional and VLM metrics for support averaging at $K = 2$. Directional values use the paired target; VLM values range from one to five.

Supports	n	Dir.-CLIP↑	Dir.-SigLIP2↑	Transfer↑	Preserve↑	No extra↑	Realism↑	Overall↑
1	352	0.766	0.791	**4.64**	4.56	**4.55**	4.05	4.37
2	352	0.800	0.817	4.62	**4.57**	4.51	**4.12**	**4.41**
4	352	**0.810**	**0.828**	4.61	4.52	4.48	3.99	4.34

The complete code-length sweep appears in Tables 9 and 10. Paired-target metrics favor shorter codes. The VLM results show the same broad decline with capacity, although the two- and four-token scores are close.

Table 9: Four support transfer across code lengths. Each entry is the mean over 352 query outputs. Inc. denotes Inception feature distance.

K	n	MSE↓	MAE↓	PSNR↑	SSIM↑	Inc.↓	CLIP↓	SigLIP2↓	DINOv2↓
2	352	**0.046**	**0.138**	**14.84**	**0.469**	**9.96**	**0.092**	**0.076**	**0.154**
4	352	0.048	0.143	14.67	0.464	10.48	0.107	0.089	0.180
8	352	0.058	0.165	13.52	0.404	12.07	0.144	0.127	0.283
16	352	0.059	0.165	13.26	0.394	12.11	0.143	0.124	0.269
32	352	0.067	0.180	12.69	0.361	13.76	0.200	0.177	0.408
64	352	0.061	0.185	12.85	0.351	15.98	0.303	0.265	0.552

Table 10: Directional and VLM transfer across code lengths with four support codes. All VLM dimensions range from one to five.

K	Dir.-CLIP↑	Dir.-SigLIP2↑	Transfer↑	Preserve↑	No extra↑	Realism↑	Overall↑
2	**0.810**	**0.828**	**4.61**	**4.52**	**4.48**	**3.99**	**4.34**
4	0.783	0.806	4.60	4.49	4.45	3.93	4.28
8	0.728	0.751	4.41	4.40	4.27	3.72	4.07
16	0.718	0.745	4.33	4.31	4.16	3.61	3.96
32	0.641	0.677	4.07	4.08	3.87	3.13	3.55
64	0.526	0.559	3.16	3.08	2.74	2.27	2.52

The transfer evaluation assigns eight queries to each of its 44 transformation identities. Table 11 reorganizes these predictions using the five display groups introduced with the dataset.

Table 11: Transfer across the five display groups at $K = 2$ with one demonstration. Each transformation contributes eight queries. Scores are first averaged within each transformation and then within each display group. Feature distances use the paired target; directional scores measure alignment with its change. The reference-free VLM score ranges from one to five.

	Paired-target distance↓			Directional alignment↑		VLM judgment↑
Transformation group	**CLIP**	**SigLIP2**	**DINOv2**	**CLIP**	**SigLIP2**	**Overall**
■ Styling	0.159	0.107	0.233	0.756	0.786	4.36
■ Lighting, color, and atmosphere	0.066	0.060	0.089	0.793	0.781	4.88
■ Viewpoint and geometry	0.110	0.081	0.242	0.727	0.779	4.58
■ Scene edits	0.105	0.127	0.331	0.702	0.713	4.50
■ Subject edits	0.105	0.093	0.165	0.784	0.807	4.22
All evaluated transformations	0.112	0.093	0.192	0.766	0.791	4.37

Consistency across transformations. We pair the code-length comparisons by transformation identity, using the same reconstruction example and eight transfer queries per identity at every K. Table 12 reports every larger code length relative to $K = 2$. At $K = 64$, reconstruction SSIM improves for 42 of 44 identities, while both directional transfer metrics decline for all 44. The endpoint tradeoff therefore extends across the evaluated transformations. The pattern is less uniform near $K = 2$, where only 20 identities show all three signs at $K = 4$.

Table 12: Paired capacity contrasts across 44 transformation identities. Transfer averages eight four-support queries per identity. Entries are mean differences (K minus 2), with pointwise 95% intervals shown as ± 1.96 bootstrap standard errors from 10,000 paired identity resamples. Joint counts identities with higher reconstruction SSIM and lower transfer alignment in both feature spaces.

K vs. 2	Δ SSIM	Δ CLIP dir.	Δ SigLIP2 dir.	Joint / 44
4	$+0.021 \pm 0.012$	-0.027 ± 0.012	-0.021 ± 0.010	20
8	$+0.057 \pm 0.014$	-0.082 ± 0.024	-0.076 ± 0.023	34
16	$+0.086 \pm 0.014$	-0.092 ± 0.026	-0.083 ± 0.024	36
32	$+0.140 \pm 0.023$	-0.169 ± 0.040	-0.151 ± 0.040	40
64	$+0.249 \pm 0.043$	-0.285 ± 0.042	-0.268 ± 0.051	42

Table 13: Mean ± sample standard deviation across evaluation outputs for the principal metrics. Transfer uses four supports and 352 queries; composition pools both orders, with n_C valid VLM responses. Best mean values are bold.

K	CLIP↓	Dir.-CLIP↑	Transfer VLM↑	n_C	Composition VLM↑
2	**0.092 ± 0.058**	**0.810 ± 0.132**	**4.34 ± 1.18**	751	3.73 ± 1.85
4	0.107 ± 0.068	0.783 ± 0.166	4.28 ± 1.22	754	**3.75 ± 1.83**
8	0.144 ± 0.093	0.728 ± 0.197	4.07 ± 1.34	758	3.61 ± 1.88
16	0.143 ± 0.080	0.718 ± 0.190	3.96 ± 1.43	759	3.69 ± 1.85
32	0.200 ± 0.109	0.641 ± 0.216	3.55 ± 1.51	760	3.70 ± 1.85
64	0.303 ± 0.108	0.526 ± 0.211	2.52 ± 1.69	759	2.85 ± 1.94

E.3 Code geometry

Table 14 collects all geometry metrics computed from the mean-pooled codes. The two-token model performs best on every reported geometry metric, while the Davies–Bouldin score is nonmonotonic at larger K.

Table 14: Code geometry over 5,268 validation pairs and 171 transformations. Centroid: leave-one-out nearest-centroid accuracy; W/B: within-to-between distance ratio; Match: Hungarian-matched cluster accuracy; Sil. and DB: silhouette and Davies–Bouldin scores (Appendix J).

K	1-NN↑	5-NN↑	Centroid↑	W/B↓	ARI↑	NMI↑	Purity↑	Match↑	Sil.↑	DB↓
2	**0.903**	**0.884**	**0.946**	**0.245**	**0.716**	**0.902**	**0.809**	**0.733**	**0.143**	**2.416**
4	0.846	0.840	0.935	0.390	0.650	0.879	0.760	0.677	0.106	2.786
8	0.828	0.796	0.889	0.412	0.531	0.826	0.674	0.583	0.091	2.782
16	0.695	0.621	0.752	0.520	0.388	0.751	0.565	0.470	0.056	2.957
32	0.616	0.565	0.703	0.495	0.332	0.718	0.516	0.417	0.054	2.770
64	0.616	0.595	0.703	0.458	0.266	0.670	0.457	0.349	0.046	2.821

Code geometry uses all 5,268 validation pairs. Across the 171 transformation identities, the median validation set contains 29 examples. Leave-one-out nearest-centroid accuracy is 94.6% when weighted by samples and 93.6% when identities are weighted equally.

E.4 Composition across both orders

For every task pair, we evaluate $[c_A; c_B]$ against sequential application in the order A then B, and $[c_B; c_A]$ against B then A. The reported composition metrics pool the outputs from both orders. Table 15 uses the corresponding sequential result as a self-reference, while Table 16 evaluates the same single-pass outputs semantically without a reference image.

Table 15: Composition agreement with sequential application, pooled over both concatenation orders. Each row contains 760 outputs. Best values are bold.

K	n	MSE↓	SSIM↑	CLIP↓	SigLIP2↓	DINOv2↓	Dir.-CLIP↑	Dir.-SigLIP2↑
2	760	**0.054**	0.535	**0.144**	**0.107**	**0.245**	0.756	0.800
4	760	0.056	0.512	0.153	0.116	0.274	0.759	0.802
8	760	0.056	0.514	0.165	0.131	0.321	0.764	0.808
16	760	0.056	**0.562**	0.153	0.131	0.321	**0.800**	**0.831**
32	760	0.070	0.530	0.206	0.180	0.442	0.769	0.801
64	760	0.055	0.466	0.307	0.274	0.680	0.653	0.682

Table 16: VLM judgments without a reference for composition, pooled over both concatenation orders. Parsing failures cause small differences in n. Scores range from one to five; best values are bold.

K	n	Edit A↑	Edit B↑	Joint↑	Preserve↑	Realism↑	Overall↑
2	751	3.75	3.75	3.74	3.75	**3.64**	3.73
4	754	3.80	3.78	**3.77**	**3.78**	**3.64**	**3.75**
8	758	3.70	3.66	3.64	3.65	3.48	3.61
16	759	3.82	3.75	3.72	3.72	3.55	3.69
32	760	**3.84**	**3.79**	3.73	3.73	3.55	3.70
64	759	2.99	2.92	2.91	2.91	2.68	2.85

E.5 Routing through the image and text-conditioning streams

The routing study changes where the code is read from the pair encoder and where it is supplied to the editor. In this notation, *text* refers to FLUX's text-conditioning stream; the model still receives no text prompt. The four routes share the same architecture, bottleneck size, training objective, and metrics, evaluated on a separate sample: 50 reconstruction pairs, 364 transfer queries with one support, and 800 single-pass compositions in the A then B order only. The text-to-text row therefore differs from the $K = 2$ rows of the code-length tables.

No routing choice dominates across the evaluations in Table 17. Using the image stream at both stages leaves transfer close to the default text-to-text route, but weakens composition. The cross-stream settings separate same-pair fidelity from reuse more clearly: image-to-text produces the strongest transfer despite lower reconstruction, whereas text-to-image gives the strongest reconstruction and

composition judgment but the least distinct code geometry. Transfer is therefore not specific to one conditioning stream, although routing affects reconstruction, composition, and code geometry.

Table 17: Routing the inferred visual code through FLUX's image and text-conditioning streams at $K = 2$, with one support per transformation. Reconstruction applies the code back to its demonstration source; transfer uses a query image; composition concatenates two independently inferred codes in the A then B order. VLM scores range from one to five. Best values are bold.

	Reconstruction	Transfer		Composition	Code geometry	
Encoder → editor	SSIM↑	Dir.-CLIP↑	VLM↑	VLM↑	1-NN↑	ARI↑
text → text	0.617	0.767	4.40	4.31	**0.903**	**0.716**
image → text	0.582	**0.794**	**4.42**	4.25	0.896	0.688
text → image	**0.640**	0.741	4.34	**4.46**	0.873	0.625
image → image	0.581	0.764	4.39	4.10	0.874	0.661

E.6 Human evaluation of natural-image composition

This study uses the $K = 2$ image-to-text route, continued from step 45,000 for 10,000 additional reconstruction steps on 480×480 images. The continuation uses batch size two and learning rate 5×10^{-5}. Sampling uses 30 denoising steps, guidance 3.0, and classifier-free guidance 3.0. Each item combines independently inferred codes with continued token positions (Appendix H). The study therefore evaluates a higher-resolution variant separately from the 256×256 code-length sweep.

The item list was fixed through an output-blind review of the demonstrations and query images before generation. Review excluded incompatible edits, inapplicable or unclear demonstrations, duplicates, and invalid assets, using the predefined validity criteria. Support and query candidates are disjoint; the two concatenation orders are equally represented. No generated output or automatic quality score was available during this selection.

We evaluate single-pass outputs through the interface in Figure 9. Annotators infer the two edits from separate source–target demonstrations and assess their application to a new query. Per-edit application averages the two edit-specific scores within each output. We first average the eligible ratings available for an output and then weight outputs equally; confidence intervals use 10,000 bootstrap replicates over outputs. Edit-specific ratings are excluded when the corresponding demonstration is marked unclear; composition and overall ratings require both demonstrations to be clear. Preservation, unintended-change, and realism ratings remain eligible in either case.

Each of the 60 outputs received two to seven ratings before the eligibility exclusions. Raters were volunteers who were informed of the purpose and the task; only their ratings were stored, without identifying information. We assessed the study, an image-rating task that collects no personal or sensitive data, and concluded that it did not require ethical review. Table 18 summarizes the human ratings.

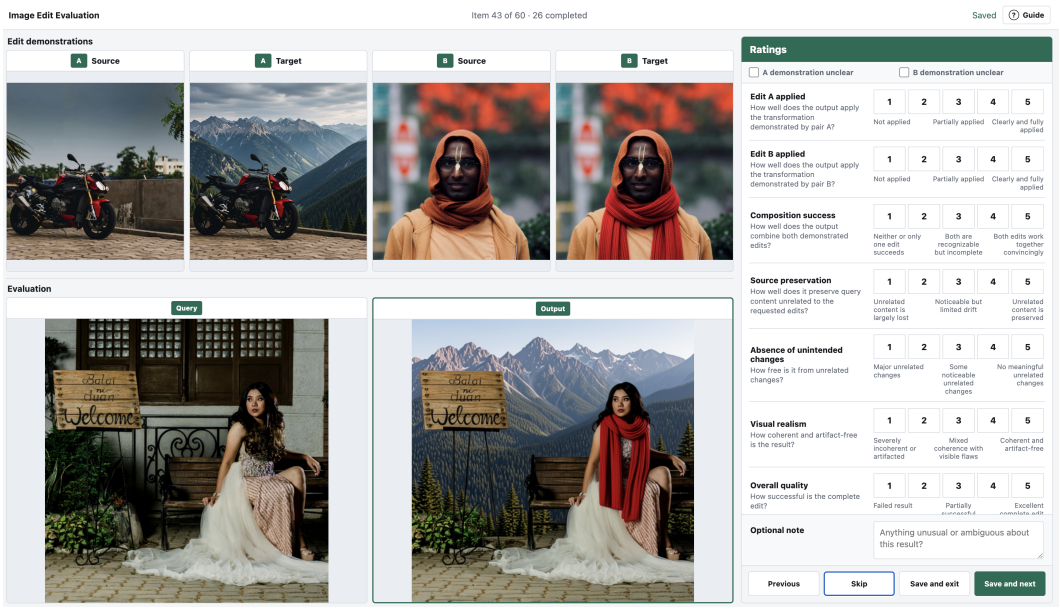

Figure 9: Human evaluation interface. Each item presents two edit demonstrations, a query, the single-pass output, and seven rating scales from one to five.

Table 18: Human evaluation of single-pass natural-image composition. Scores range from one to five; higher is better.

Dimension	Mean	95% CI
Per-edit application	3.86	[3.66, 4.05]
Composition success	3.77	[3.53, 4.00]
Source preservation	4.35	[4.22, 4.47]
Absence of unintended changes	4.22	[4.07, 4.36]
Visual realism	4.26	[4.11, 4.41]
Overall quality	4.06	[3.88, 4.23]

F TRANSFER ON RELATION

Here we test whether the two-token bottleneck transfers on a second collection under a conventional one-support protocol. We train on Relation, the collection released with RelationAdapter (Gong et al., 2025), and report the released RelationAdapter, LoRWeB (Manor et al., 2026), and Edit Transfer (Chen et al., 2025) pipelines as reference points. The systems differ in training data, backbones, textual conditioning, and output resolution.

We reconstruct a deterministic split from the public Relation files, reserving five queries from each of 208 tasks. This leaves 15,238 pairs for training and 1,040 held-out transfers, with no overlap in pair identifiers or resolved image paths. Our model is trained on this split. Comparable training manifests are unavailable for the released RelationAdapter and LoRWeB checkpoints, which were trained on Relation252K. At inference, our model receives only the demonstration pair and query image, whereas the reference pipelines also receive the public edit instruction. Our model uses the image-to-text route (Appendix E.5) trained for 45,000 steps with the remaining settings of Table 3; we use the final checkpoint without validation selection.

We evaluate the released Relation252K checkpoints for RelationAdapter and LoRWeB and the original Edit Transfer checkpoint; in Table 19, (reprod.) marks the released LoRWeB checkpoint trained on Relation252K and (orig.) the original Edit Transfer checkpoint. Delta-Adapter (Chen et al.,

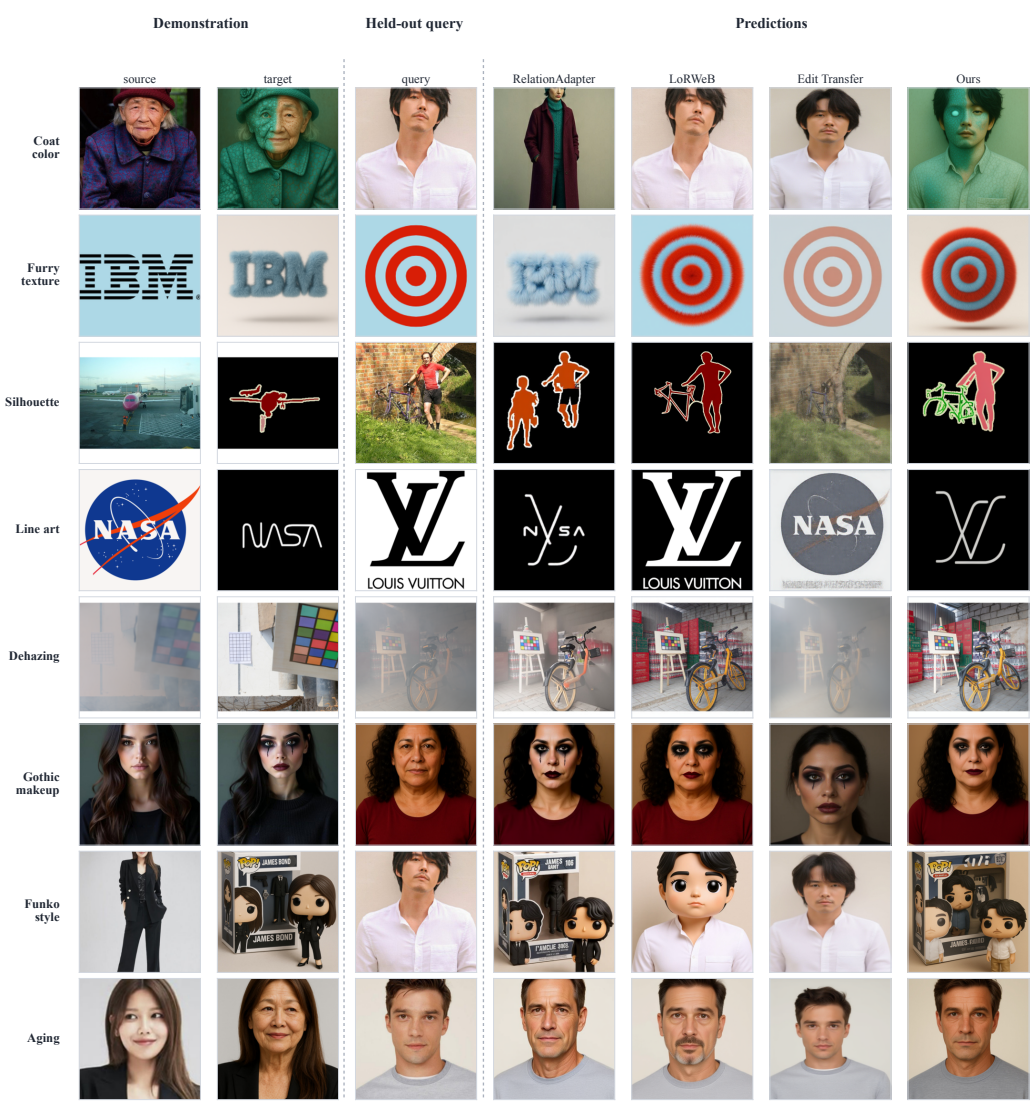

Figure 10: **One-support transfer on Relation.** Each row contains the demonstration, a held-out query, and four predictions, with our method in the final column. The paired edited reference used by Table 19 is omitted because it is one valid realization rather than a unique target. Our model infers its code from images alone; the released pipelines also receive the public edit instruction. All eight cases are shown. Panels are center-cropped; metrics use the full outputs.

2026) is omitted because neither its trained checkpoint nor its exact evaluation manifest was publicly available.[1]

Table 19 reports paired-reference CLIP-I, directional CLIP, and paired-reference LPIPS (Zhang et al., 2018). CLIP metrics use ViT-L/14. LPIPS uses AlexNet features after resizing each prediction to the reference dimensions. Relation supplies one edited reference for each query, and other outputs can be equally valid. Uncertainty is the standard error across 208 task means, which keeps the five queries sharing a task and demonstration within the same evaluation unit.

[1]Our public reconstruction contains 208 tasks and 1,040 held-out queries, whereas Delta-Adapter reports 218 tasks.

Table 19: One-support transfer on the public Relation split. Values are mean ± standard error across 208 task means, with five held-out queries per task.

	CLIP cosine ↑		LPIPS ↓
Pipeline	Reference	Direction	Reference
RelationAdapter	0.822 ± 0.006	0.649 ± 0.012	0.462 ± 0.012
LoRWeB (reprod.)	0.814 ± 0.006	0.585 ± 0.013	0.511 ± 0.012
Edit Transfer (orig.)	0.693 ± 0.006	0.336 ± 0.008	0.634 ± 0.012
Ours	0.884 ± 0.004	0.765 ± 0.009	0.510 ± 0.009

The two-token bottleneck also transfers on Relation under a conventional one-support protocol and without access to the edit instruction. Figure 10 shows all eight qualitative cases, spanning color, texture, structural, restoration, and attribute edits. The set includes the aging case, where our prediction is less faithful than some of the released pipelines.

G An Illustrative Model of the Reconstruction–Transfer Tradeoff

In the capacity sweep, a wider code reconstructs its own pair more faithfully while a narrower code transfers more accurately to a new image. We give an idealized account of how this can occur by modeling recurring and support-specific variation through a capacity-limited representation. The analysis concerns the reconstruction–transfer tradeoff; token count, diffusion optimization, and code composition remain outside the model.

Let T denote a transformation, $S = (X_s, Y_s)$ a support pair, and (X_q, Y_q) a query pair governed by the same transformation. Outputs take values in a finite-dimensional Hilbert space $\mathcal{H}$, and all variables below are square-integrable. We assume that the support carries no information about query-specific variation once the transformation and query input are known:

$$Y_q \perp\!\!\!\perp S \mid (T, X_q). \tag{5}$$

Define the reusable action

$$M = \mathbb{E}[Y_q \mid T, X_q]. \tag{6}$$

Under squared loss, M is the Bayes-optimal output when the transformation and query input are retained but the particular query realization is not.

An encoder obtains Z only from S, so that Z is conditionally independent of (T, X_q, Y_q) given S. An editor produces $\widehat{Y}_q = g(Z, X_q, W)$, where W denotes independent editor randomness. Set

$$m_S = \mathbb{E}[M \mid S, X_q], \qquad m_Z = \mathbb{E}[M \mid Z, X_q]. \tag{7}$$

The first conditional mean is the best reusable action recoverable from the full support; the second is what remains available after compression.

Proposition 1 (support–query decomposition). Every encoder and editor satisfying the conditions above obeys

$$\begin{aligned}\mathbb{E}\|Y_q - \widehat{Y}_q\|^2 = {} & \underbrace{\mathbb{E}\|Y_q - M\|^2}_{V_{\text{query}}} + \underbrace{\mathbb{E}\|M - m_S\|^2}_{V_{\text{support}}} \\ & + \underbrace{\mathbb{E}\|m_S - m_Z\|^2}_{V_{\text{compression}}} + \underbrace{\mathbb{E}\|m_Z - \widehat{Y}_q\|^2}_{V_{\text{editor}}}.\end{aligned} \tag{8}$$

Proof. Write the prediction error as the sum of

$$R_0 = Y_q - M, \quad R_1 = M - m_S, \quad R_2 = m_S - m_Z, \quad R_3 = m_Z - \widehat{Y}_q.$$

Equation 5 gives $\mathbb{E}[R_0 \mid T, S, Z, X_q, W] = 0$. Because Z is obtained only from S and W is independent, $\mathbb{E}[R_1 \mid S, Z, X_q, W] = 0$. The encoder conditional independence and the tower

property give $\mathbb{E}[m_S \mid Z, X_q] = m_Z$, and hence $\mathbb{E}[R_2 \mid Z, X_q, W] = 0$. Each residual is therefore orthogonal in L^2 to every subsequent residual. Expanding the squared norm of $R_0 + R_1 + R_2 + R_3$ proves Equation 8. □

The first two terms are fixed by the data distribution and support protocol. The compression term is zero exactly when m_S is recoverable from (Z, X_q). The code may therefore discard information needed to reproduce the particular support pair without increasing Bayes transfer risk. The decomposition does not, by itself, guarantee that reconstruction training will select such a code. The following special case gives one sufficient mechanism.

G.1 A LINEAR CAPACITY THRESHOLD

Let a demonstrated transformation be

$$\tau_{A,U} = \sum_{i=1}^{p} A_i \phi_i + \sum_{j=1}^{q} U_j \psi_j, \tag{9}$$

where p and q are positive integers and the directions are orthonormal. The coordinates A_i describe a recurring transformation, while U_j describe details specific to its realization in the support. Assume that all coefficients are independent and zero mean, with

$$\mathrm{Var}(A_i) = \alpha_i, \qquad \mathrm{Var}(U_j) = \beta_j, \qquad \alpha_1 > \cdots > \alpha_p > \beta_1 > \cdots > \beta_q > 0. \tag{10}$$

A query shares A but has independent realization coordinates U' drawn from the same distribution as U. Let P_k be the reconstruction map of a population-optimal rank-k linear autoencoder, and define

$$\mathcal{R}_{\mathrm{rec}}(k) = \mathbb{E}\|\tau_{A,U} - P_k \tau_{A,U}\|^2, \tag{11}$$

$$\mathcal{R}_{\mathrm{transfer}}(k) = \mathbb{E}\|\tau_{A,U'} - P_k \tau_{A,U}\|^2. \tag{12}$$

Proposition 2 (reconstruction–transfer threshold). For $0 \le k < p$,

$$\mathcal{R}_{\mathrm{rec}}(k) = \mathcal{R}_{\mathrm{transfer}}(k) = \sum_{i=k+1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j. \tag{13}$$

At $k = p + r$, where $0 \le r \le q$,

$$\mathcal{R}_{\mathrm{rec}}(p+r) = \sum_{j=r+1}^{q} \beta_j, \qquad \mathcal{R}_{\mathrm{transfer}}(p+r) = \sum_{j=1}^{q} \beta_j + \sum_{j=1}^{r} \beta_j. \tag{14}$$

Thus reconstruction error decreases whenever $0 \le k < p + q$, whereas transfer error is uniquely minimized at $k = p$.

Proof. The optimal rank-k linear autoencoder projects onto the leading covariance directions (Baldi & Hornik, 1989). By Equation 10, all recurring directions enter before any realization-specific direction. For $k < p$, both reconstruction and transfer omit the same recurring coordinates and all q realization coordinates, giving Equation 13. At $k = p + r$,

$$P_{p+r} \tau_{A,U} = \sum_{i=1}^{p} A_i \phi_i + \sum_{j=1}^{r} U_j \psi_j.$$

The reconstruction error contains only the omitted support coordinates. For transfer, each retained private coordinate contributes $\mathbb{E}(U'_j - U_j)^2 = 2\beta_j$, while each omitted one contributes $\mathbb{E}(U'_j)^2 = \beta_j$. This yields Equation 14; strict positivity of the variances gives the unique minimum. □

Below the threshold p, additional capacity recovers missing parts of the recurring transformation and helps both objectives. Above it, each private direction improves support reconstruction but substitutes a detail from the demonstration for the independent detail required by the query. The capacity sweep shows the same qualitative pattern, with reconstruction improving while transfer worsens once the channel begins to carry pair-specific variation.

H CONTROLLED COMPOSITION: FULL PROTOCOL AND ORDER SENSITIVITY

The main paper focuses on compatible pairs, for which both code orders share one exact target. Here we describe the full protocol and add different-color overlaps: both rules can address the same object, and the last rule determines its color. These cases have two ordered targets and test whether swapping the code blocks controls execution order.

Each scene contains sixteen colored objects on a 4×4 grid. An atomic transformation selects objects by shape, size, or spatial region and assigns a new color. Symbolic scene and transformation records are used to construct targets and score predictions.

Table 20: Configuration of the controlled composition study. Unlisted model and optimization settings follow Table 3.

Parameter	**Value**
Code configuration	
Code length K	2
Code route	Image stream in the pair encoder; text-conditioning stream in the editor
Training data	
Atomic	1,050,000 atomic reconstruction pairs
Combined	1,050,000 atomic and 300,000 joint reconstruction pairs
Combined training	One compressed code reconstructs each joint target; concatenated codes are never used during training
Held-out compositions	Evaluation transformation pairs are absent from Combined training
Training configuration	
Global batch size	24
Optimizer steps	43,750 (Atomic); 50,000 (Combined)
Shared settings	Architecture, LoRA, optimizer, image resolution, and regularization follow Table 3
Evaluation configuration	
Sampling	25 steps; CFG 1.0; denoiser guidance 3.0; seed 0
Evaluation set	192 held-out transformation pairs; four query scenes per pair; both orders

We train two checkpoints (Table 20) to test whether one-pass composition depends on the editor's ability to render joint targets. We call them *Atomic* and *Combined*. Atomic is trained on 42 transformations over 25,000 source scenes, for 1.05 million reconstruction pairs; every target follows one selection rule and one color assignment. Combined adds 300,000 reconstruction pairs whose targets combine two atomic rules. Each such target is still compressed into a single code, and the training pairs use different transformation combinations from evaluation. Neither checkpoint is trained by concatenating two codes.

Evaluation pairs atomic transformations that occur individually during training. We use 192 held-out pairs and four new query scenes per pair, and evaluate both orders. The three pair types contain 64 pairs each. *Disjoint* pairs act on non-overlapping sets of objects. *Same-color overlap* pairs may address the same objects but assign the same color. These first two types form the compatible subset in Table 2. *Different-color overlap* pairs assign different colors to at least one shared object and are order sensitive. All query scenes are distinct from the training, validation, and support pools.

For each pair, one demonstration supplies c_A and another supplies c_B. Addition and averaging combine the codes elementwise and retain length $K = 2$. Concatenation produces a four-vector sequence. The repeated-position variant reuses the original two position identifiers for both blocks, while the continued-position variant assigns the second block the next two positions. Here, position refers to the token identifiers from which the editor constructs its rotary positional embedding (RoPE): the former concatenates the identifiers produced for two separate codes, whereas the latter generates them for the complete four-token sequence. Unless stated otherwise, concatenation uses continued RoPE positions. Sequential execution applies c_A and c_B in two separate denoising passes.

Exact-scene success requires all sixteen object colors to match the target. Edited-object accuracy measures whether the requested changes are made, and preservation accuracy measures whether all other objects remain unchanged. For different-color overlaps, order accuracy is the fraction of predictions whose object-color accuracy is higher against the requested-order target than against the reverse-order target; ties count as incorrect.

Table 21: Full controlled benchmark pooled across disjoint, same-color overlap, and order-sensitive different-color overlap pairs. Values are percentages over both orders (1,536 outputs per procedure). Sequential execution uses two denoising passes and is reported separately.

Atomic

Inference	Exact	Edited	Preserved
Add: $c_A + c_B$	6.3	43.3	94.0
Average: $(c_A + c_B)/2$	5.1	40.4	94.6
Concat., repeated positions	7.4	**48.4**	**97.3**
Concat., continued positions	**8.1**	47.4	97.2
Sequential, two passes	92.8	97.8	99.1

Combined

Inference	Exact	Edited	Preserved
Add: $c_A + c_B$	15.7	53.0	96.8
Average: $(c_A + c_B)/2$	16.3	52.5	96.4
Concat., repeated positions	**75.0**	**93.7**	**98.7**
Concat., continued positions	74.5	93.6	98.6
Sequential, two passes	97.1	99.6	99.7

Table 21 pools the compatible and order-sensitive problems. The Atomic checkpoint reaches 92.8% exact-scene success with two sequential passes, but 8.1% when continued-position codes are concatenated and decoded once. With Combined, the corresponding one-pass result rises to 74.5% across all three pair types and 90.8% on the compatible subset reported in the main paper. Both checkpoints execute the codes accurately in two passes, while accurate one-pass concatenation occurs only with Combined. Figure 6 shows examples from Combined; Figure 11 shows examples produced by Atomic.

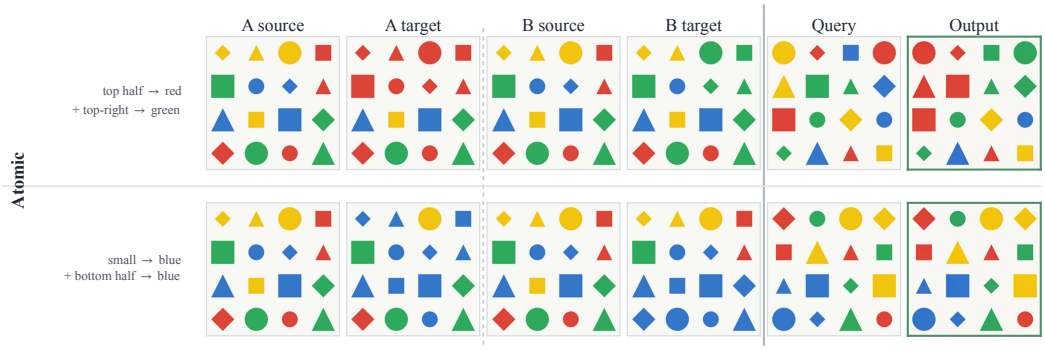

Figure 11: **Exact one-pass compositions under Atomic training.** Each example begins with two independent visual demonstrations, separated from the application columns by the vertical rule. These demonstrations produce c_A and c_B, which are concatenated only at inference and applied to the query. Both outputs shown here satisfy the indicated joint rules exactly.

Table 22: Results by pair type. The first three columns report exact-scene success (%); order accuracy applies only to different-color overlaps. Each cell aggregates 512 ordered outputs.

Inference	Disjoint	Same-color	Different-color	Order acc.
Atomic, concat.	7.4	13.1	3.7	45.3
Combined, concat.	96.5	85.2	42.0	45.5
Atomic, sequential	94.7	92.6	91.2	95.1
Combined, sequential	99.0	96.5	95.9	100.0

The pair-type breakdown (Table 22) separates joint control from ordered execution. Continued-position concatenation with Combined reaches 96.5% on disjoint pairs and 85.2% on same-color

overlaps. On different-color overlaps, exact-scene success falls to 42.0% and order accuracy is 45.5%, so swapping the token blocks does not reliably select the corresponding ordered target. Applying the same codes in two passes instead reaches 95.9% exact-scene success and 100% order accuracy. Concatenation therefore combines compatible changes but does not implement general noncommutative function composition. Figure 12 illustrates the distinction for one order-sensitive pair.

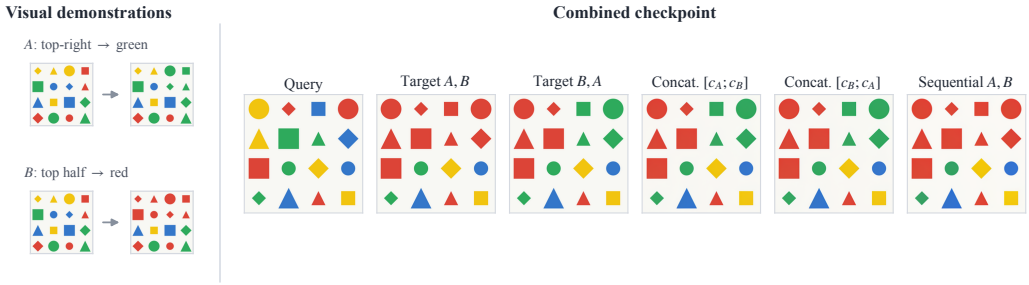

Figure 12: **Concatenation does not define a temporal order.** This example uses the Combined checkpoint. The visual pairs on the left are the independent demonstrations from which c_A and c_B are inferred. Reversing the two code blocks leaves the one-pass prediction unchanged, while applying the same codes in two passes preserves the requested $A \to B$ order.

As Section 6 notes, many nominally single natural-image transformations already require coordinated changes, such as a costume edit that alters a shirt, trousers, and accessories together, so the natural-image editor encounters multi-part changes during ordinary reconstruction.

I Additional Natural-Image Examples

I.1 Single-Transformation Transfer

Figures 13 and 14 show the remaining curated single-transformation cases; the three examples in Figure 3 are not repeated. The gallery order was fixed with seed 20260820. Each row contains the visual demonstration, a query image, and the editor output. As in the main figure, the paired dataset realization used by reference-based metrics is omitted.

I.2 Composition

Figures 15–18 show one query for each remaining curated transformation pair; the four main paper cases are not repeated. The gallery was shuffled with seed 20260820 after retaining at most one query per pair and, when possible, avoiding reuse of a query source. Each row shows the two visual demonstrations, the held-out query, and the one-pass output from their concatenated codes.

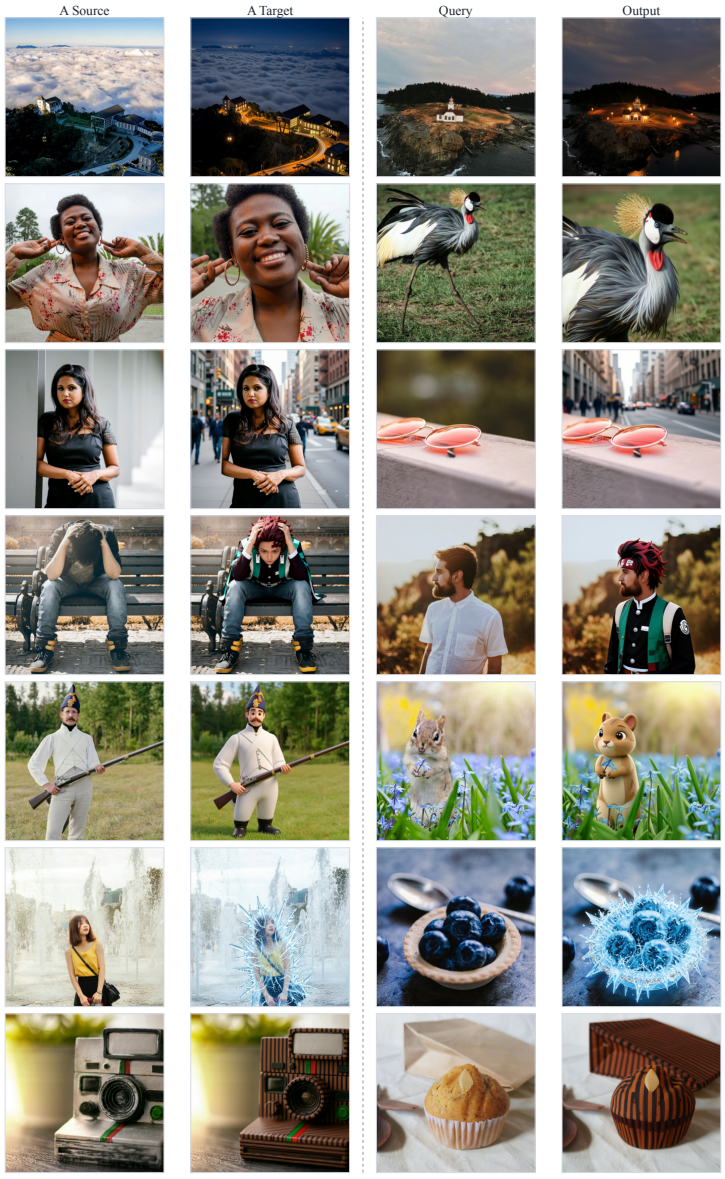

Figure 13: Additional single-transformation transfers, shown in fixed random order (part 1 of 2). Each row contains the demonstration, query image, and editor output.

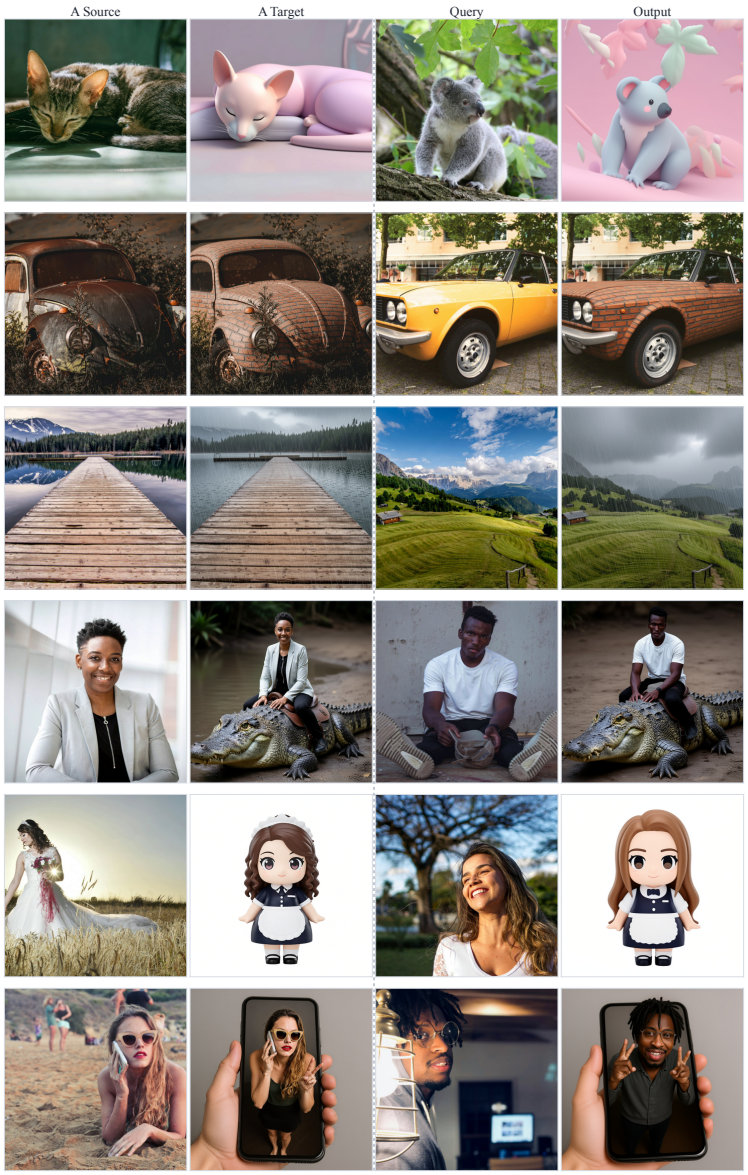

Figure 14: Additional single-transformation transfers (part 2 of 2).

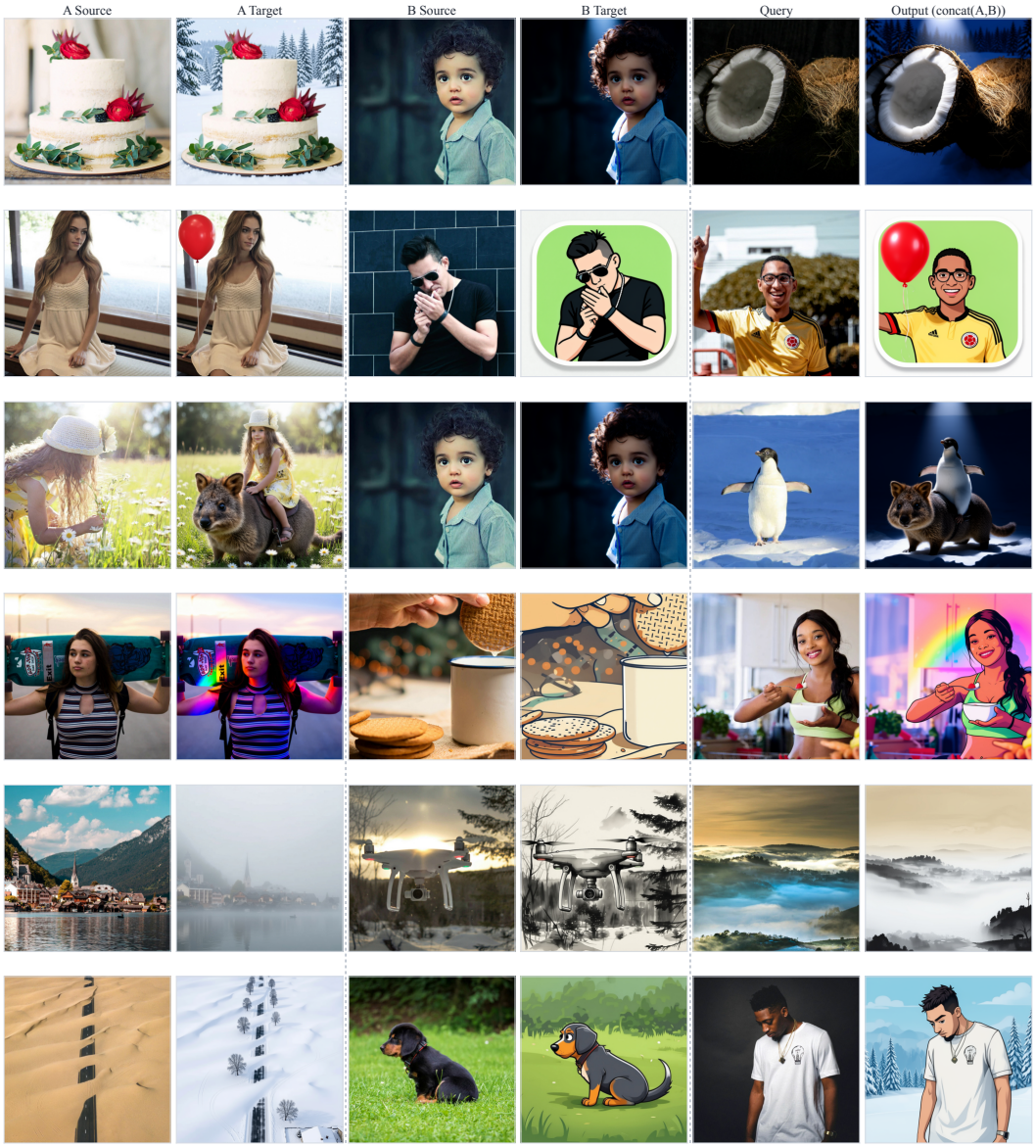

Figure 15: Additional one-pass compositions, shown in fixed random order (part 1 of 4). Each row shows the two demonstrations, the held-out query, and the one-pass output.

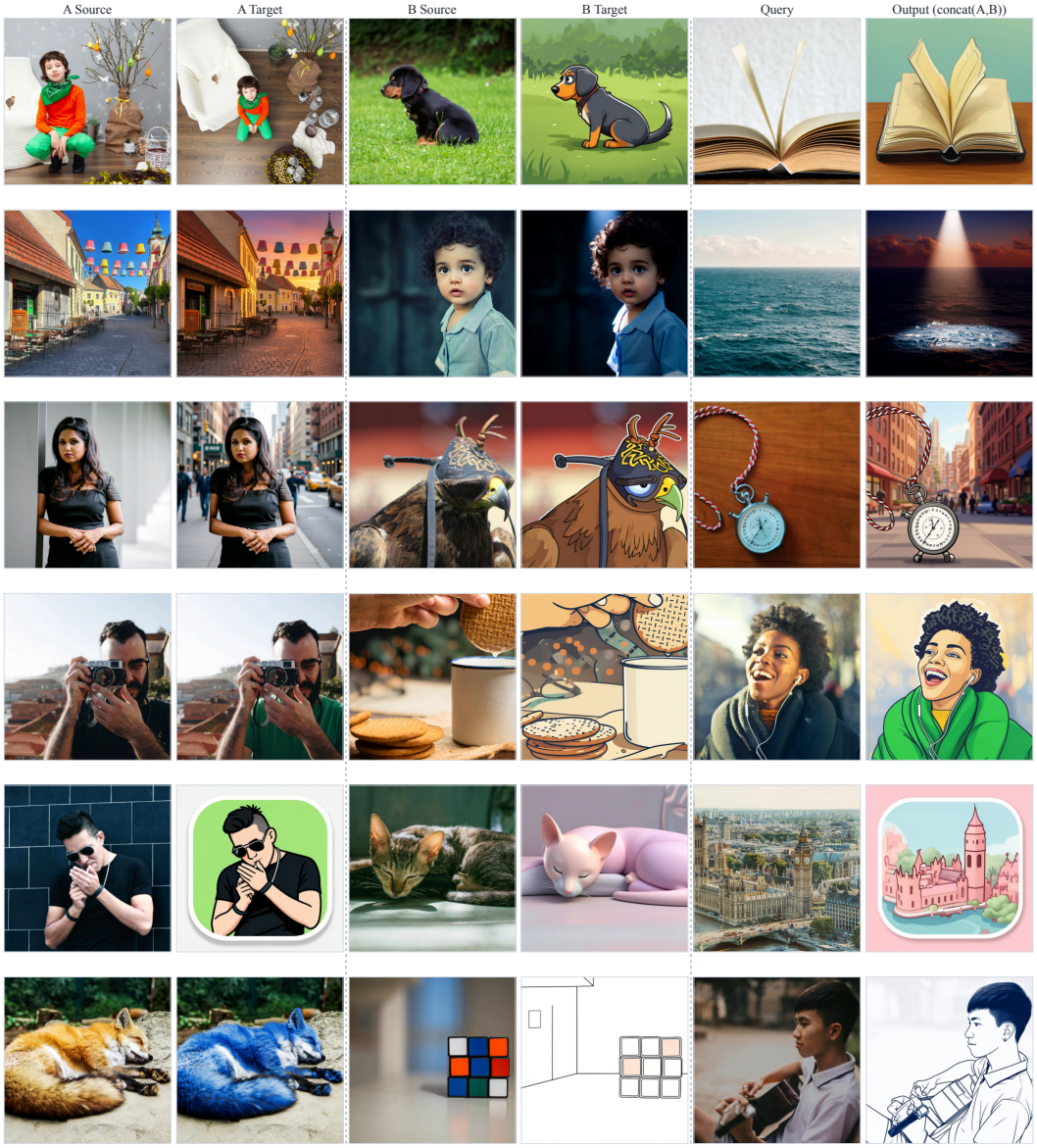

Figure 16: Additional one-pass compositions (part 2 of 4).

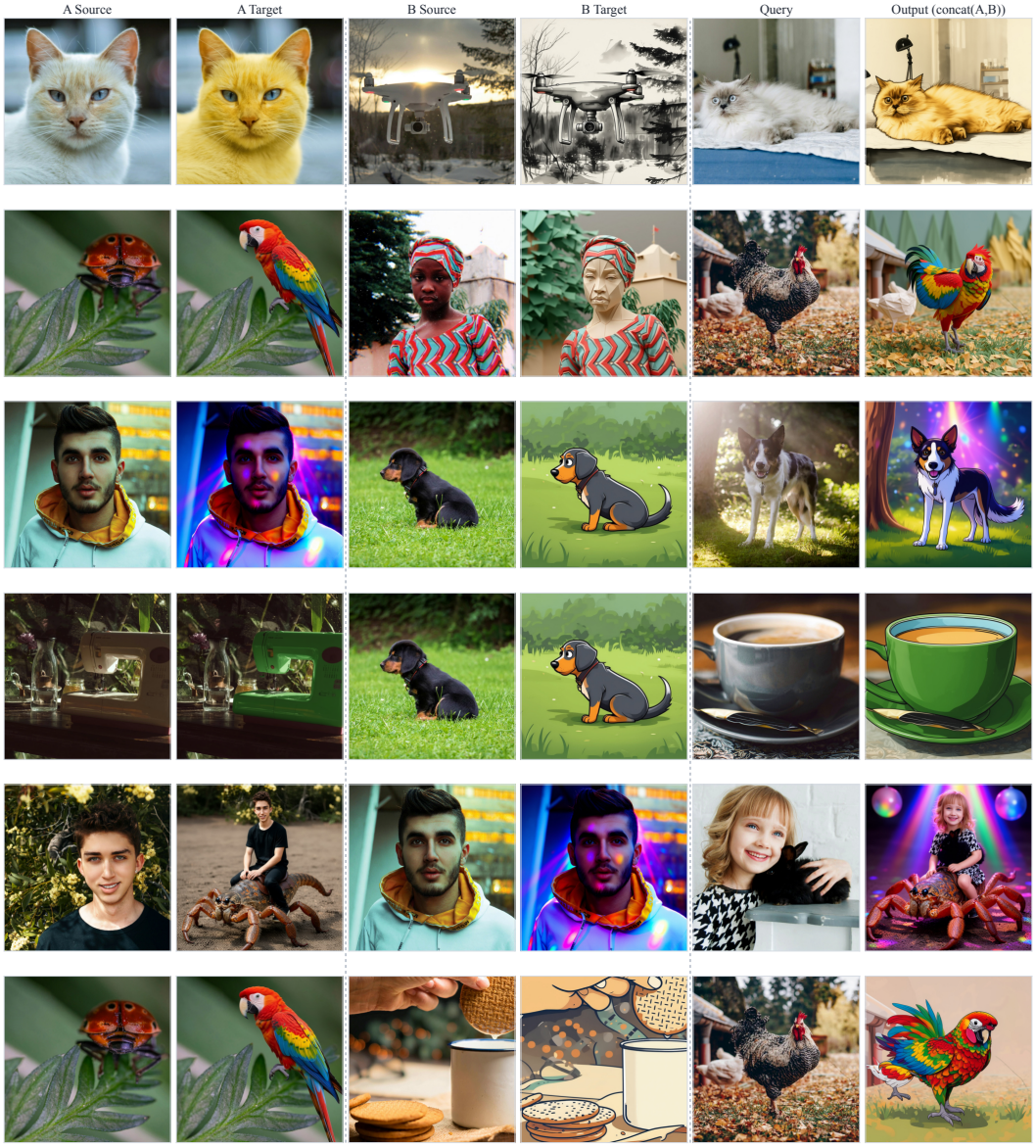

Figure 17: Additional one-pass compositions (part 3 of 4).

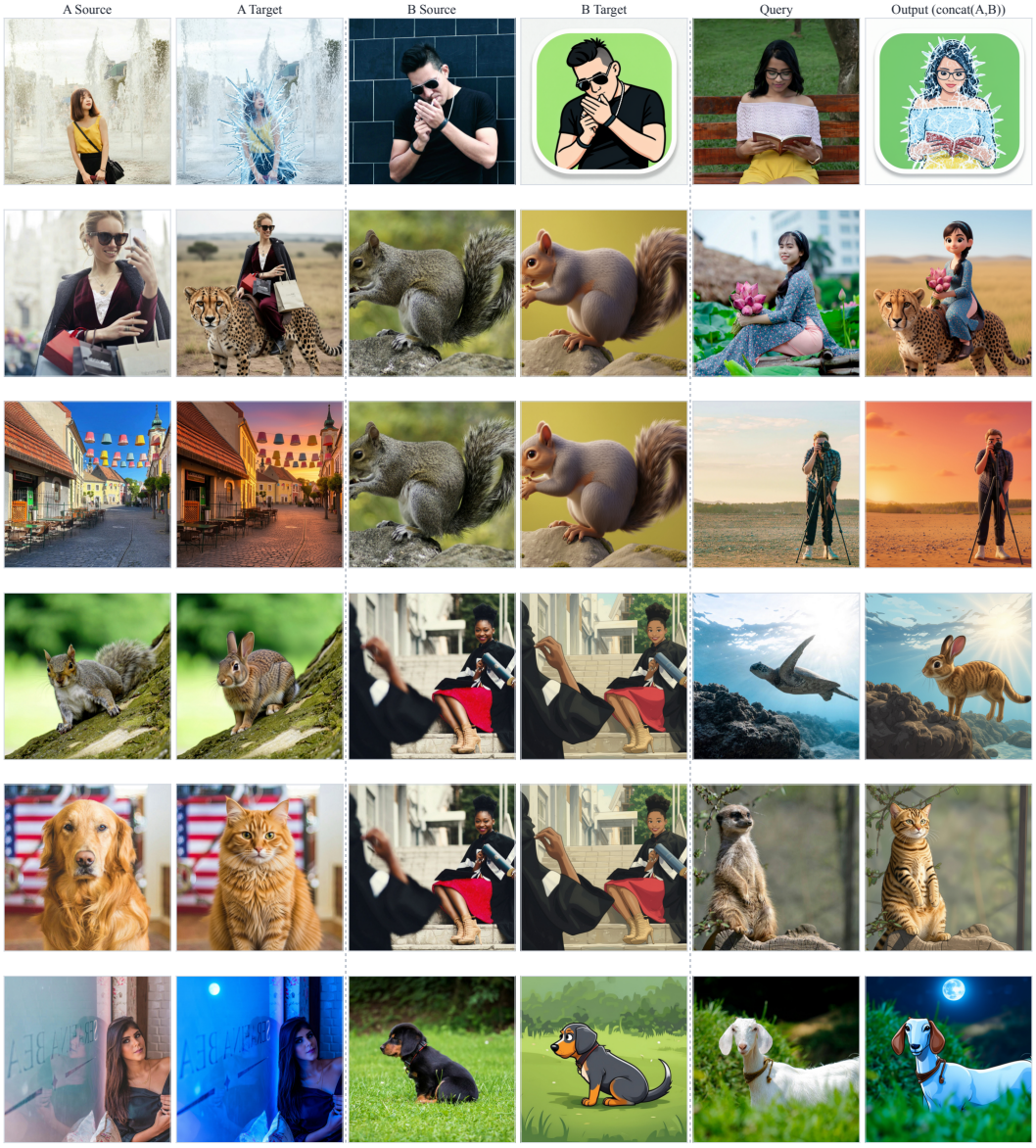

Figure 18: Additional one-pass compositions (part 4 of 4).

J METRIC DEFINITIONS

Image agreement. MSE and MAE measure pixel error; PSNR and SSIM (Wang et al., 2004) capture reconstruction fidelity. Inception L2 is Euclidean feature distance, while CLIP, SigLIP2, and DINOv2 use cosine distance. Lower is better for every distance; higher is better for PSNR and SSIM. Tables report means, and Table 13 additionally reports per-example standard deviations for the principal metrics.

Directional agreement. For a feature extractor f, query x, prediction $\hat{y}$, and reference y, the directional score is

$$s_{\mathrm{dir}} = \frac{\langle f(\hat{y}) - f(x),\, f(y) - f(x) \rangle}{\|f(\hat{y}) - f(x)\|_2 \, \|f(y) - f(x)\|_2}. \tag{15}$$

The transfer reference is the paired target after a VAE round trip. The composition reference is the result of sequential application in the same order. Higher values indicate better alignment of the two feature changes.

Code geometry. For validation pair i, let $c_{it} \in \mathbb{R}^d$ be code token t and $y_i \in \{1, \dots, R\}$ its transformation identity. We first mean-pool the K tokens,

$$z_i = \frac{1}{K} \sum_{t=1}^{K} c_{it}, \qquad d(z_i, z_j) = 1 - \frac{z_i^\top z_j}{\|z_i\|_2 \|z_j\|_2}. \tag{16}$$

If $\mathcal{N}_k(i)$ contains the k nearest other codes under d, the reported neighborhood precision is

$$P@k = \frac{1}{nk} \sum_{i=1}^{n} \left| \{ j \in \mathcal{N}_k(i) : y_j = y_i \} \right|. \tag{17}$$

The cardinality counts same-transformation neighbors. Thus 1-NN accuracy and 5-NN precision are the cases $k = 1$ and $k = 5$.

Let μ_r be the centroid of transformation r, and let $\mu_r^{(-i)}$ omit z_i when $r = y_i$. Nearest-centroid accuracy and the within-to-between ratio are

$$A_{\mathrm{cent}} = \frac{1}{n} \left| \left\{ i : y_i = \arg\min_r d\Big(z_i, \mu_r^{(-i)}\Big) \right\} \right|, \tag{18}$$

$$\mathrm{W/B} = \frac{\frac{1}{n} \sum_i d(z_i, \mu_{y_i})}{\frac{2}{R(R-1)} \sum_{r<s} d(\mu_r, \mu_s)}. \tag{19}$$

A lower W/B value indicates that codes are compact within transformations relative to the separation between transformation centroids.

We separately fit Euclidean k-means with $R = 171$ clusters, seed 0, and ten initializations. Let u_i be the assigned cluster, n_{ur} the number of samples in cluster u with label r, $a_u = \sum_r n_{ur}$, and $b_r = \sum_u n_{ur}$. The label-agreement measures are

$$\mathrm{Purity} = \frac{1}{n} \sum_u \max_r n_{ur}, \qquad \mathrm{Match} = \frac{1}{n} \max_\pi \sum_u n_{u,\pi(u)}, \tag{20}$$

$$\mathrm{NMI} = \frac{I(U;Y)}{\sqrt{H(U)H(Y)}}, \qquad \mathrm{ARI} = \frac{S - E}{\frac{1}{2}(A + B) - E}, \tag{21}$$

where $S = \sum_{u,r} \binom{n_{ur}}{2}$, $A = \sum_u \binom{a_u}{2}$, $B = \sum_r \binom{b_r}{2}$, and $E = AB/\binom{n}{2}$. Match maximizes over one-to-one cluster-to-label assignments. Finally, if d_i^{in} is the mean distance from z_i to its own cluster, d_i^{out} the smallest mean distance to another cluster, ν_u a cluster centroid, and s_u its mean within-cluster distance, then

$$\mathrm{Sil} = \frac{1}{n} \sum_i \frac{d_i^{\mathrm{out}} - d_i^{\mathrm{in}}}{\max(d_i^{\mathrm{in}}, d_i^{\mathrm{out}})}, \qquad \mathrm{DB} = \frac{1}{R} \sum_u \max_{v \neq u} \frac{s_u + s_v}{\|\nu_u - \nu_v\|_2}. \tag{22}$$

Higher is better for the agreement and silhouette measures; lower is better for W/B and Davies–Bouldin.

VLM judgment. We use Qwen3.5-9B as a reference-free semantic judge. For transfer, it sees the support source–target pairs, query source, and prediction, and gives 1–5 scores for edit transfer, preservation, absence of unintended changes, realism, and overall quality. For composition, it sees both sets of support pairs, the query source, and the single-pass prediction, scoring edits A and B separately, their joint success, and the same preservation and quality criteria. Neither the paired query target nor the sequential composition output is shown. Responses use a fixed JSON format; unparseable responses are excluded from the means.