

A Details of Benchmark and Metrics

As the pioneering work investigating the unified virtual try-on task, we construct a comprehensive evaluation benchmark named *OmniTry-Bench*, accompanied by six dedicated metrics to systematically assess the quality of synthesized try-on images.

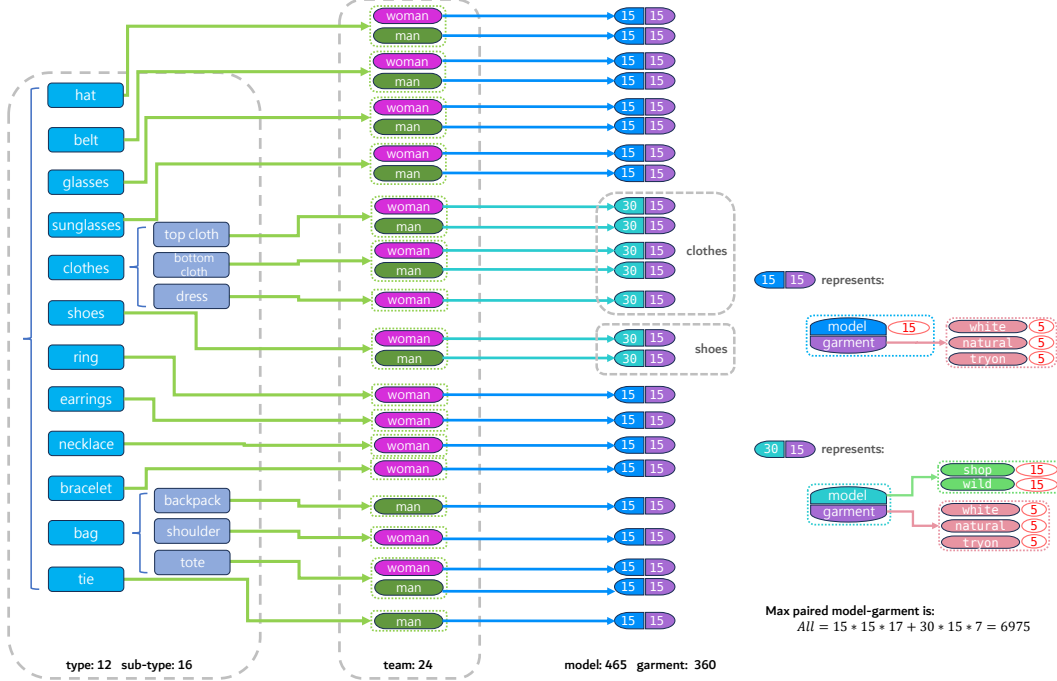


Figure 7: The visualization of the *OmniTry-Bench* constitution.

A.1 Constitution of Benchmark

As the Figure 7, we gather evaluation samples within 12 common types of wearable objects, which can be summarized into 4 major classes: (i) *clothes* consisting of top, bottom and full-body garments, (ii) *shoes* in common styles, (iii) *jewelries*, including bracelets, earrings, necklaces and rings, (iv) *accessories*, including bags, belts, hats, glasses, sunglasses and ties.

We consider detailed sub-types if necessary, such as the class *bag* consisted of the backpack, shoulder and tote bags. *Clothes* are divided into top cloth, bottom cloth, and dress. Each sub-type contains two gender groups (woman and man), with the exceptions that *jewelries* and *dress* exclusively contain woman samples, while *tie* contains only man samples.

Each gender group includes 15 model images, where the garments are categorized into three settings: white background, natural background, and try-on setting. Every garment setting include 5 images. Following previous work’s categorization of virtual try-on scenarios into *in-shop* and *in-the-wild*, we further divide the model images for *clothes* and *shoes* into 15 shop-style and 15 wild-style samples per gender group, resulting in 30 model images per sub-type.

The benchmark predominantly sources images from public repositories (Pexels²), supplemented with brand website materials and social media content under compliant data usage protocols.

Pairing Strategy. For each gender group, we establish combinatorial pairs between model and garment images through:

- *Maximum Pair Calculation:* $max_pairs = 15 \times 15 \times 17 + 30 \times 15 \times 7 = 6,975$ pairs, where 17 and 7 denote model settings counts for regular and style -specific categories respectively.

²<https://www.pexels.com>

- *Sampled Pair Selection*: $selected_pairs = 15 \times 15 \times 24 = 360$ paired samples, constrained by single-use garment policy and balanced sampling (15 models per clothes/shoes type, include 7 shop-style and 8 wild-style).

Overall, our experiments are all evaluated on the selected benchmark contains 360 pairs of images

A.2 Evaluation Metrics

As discussed before, the objectives of try-on can be divided into three aspects. Since there is no ground-truth result in mask-free setting, we redesign the metrics as follows:

Object Consistency: We crop the objects from the try-on and object images via masking, then perform white-background normalization on the extracted objects. We compute the visual similarity using DINO [3] and CLIP [49] visual encoders, with metrics denoted as M-DINO and M-CLIP-I. As these metrics measure cosine similarity in the embedding space, their values range in $[-1, 1]$ where higher values indicate better object preservation. The M-DINO scores generally exhibit lower values than M-CLIP-I, as DINO-extracted features are more sensitive to geometric variations compared to CLIP’s semantic-aligned embeddings. Our experiments quantitatively validate this behavior across different object categories. This discrepancy stems from their distinct learning objectives:

- **M-DINO** [3]: Learns dense local features through self-supervised distillation, emphasizing spatial consistency of object parts. Then compute the cosine similarity of two features.
- **M-CLIP-I** [49]: Optimizes global semantic alignment between object images, prioritizing category-level coherence. Then compute the cosine similarity of two features. Then compute the cosine similarity of two object features.

Person Preservation: We extract the person regions by cropping try-on and original person images, masking the target object areas with black pixels. We then compute spatial-aligned similarity between these aligned image pairs using two complementary metrics:

- **SSIM** (Structural Similarity Index) [57]: Measures structural, luminance, and contrast similarity between images. The metric ranges in $[-1, 1]$ with values approaching 1 indicating higher structural consistency.
- **LPIPS** (Learned Perceptual Image Patch Similarity) [65]: Computes deep feature differences using pretrained VGG networks, better aligning with human perception than traditional metrics. Its values lie in $[0, 1]$ where lower scores denote better preservation quality.

Object Localization: We propose a dual-strategy evaluation framework to assess spatial rationality through complementary approaches:

- **G-Accuracy**: Quantifies detection reliability using GroundingDINO [36] with the following implementation protocol: Invoke *predict_with_classes* API with target object categories as *classes* parameter. Configure detection thresholds: *box_threshold* = 0.25 (bounding box confidence) and *text_threshold* = 0.25 (text-image alignment). Last, calculate success rate as total test cases correct detections.
- **CLIP-I**: Evaluates semantic alignment through multi-modal similarity measurement: Generate descriptive prompts via Qwen2 [1] MLLM. Compute CLIP [49] embedding similarity between try-on images and generated text. Normalize scores to $[0, 1]$ range using min-max scaling.

The final prompt template is formally defined as follows:

```
"""Generate a detailed description of a composite image by combining
elements from the two provided images:
1. Image 1: The model's appearance (pose, clothing, facial features), background and style
2. Image 2: Only the <{garment_class}>, without any other infos
(e.g., background, model)
Describe the synthesized image with the model wearing the {garment_class}, in 65 words. Only describe the final imagined scene, without the detail or information of composite. The main description is from
```

Image 1. Briefly and shortly describe the {garment_class} in 6 words, no details needed. No words like (e.g., from the Image 2). If {garment_class} is cloth or dress , the model from the Image 1, replace with the {garment_class} from Image 2, no words like (replace the hair/shirt), using "wear" the {garment_class}.

Examples outputs:

- "A young woman standing in a studio with a white background. She is wearing a denim dress with a button-down collar and long sleeves. The dress is knee-length and falls above her knees. The woman is also wearing black ankle boots with a pointed toe and a low heel. She has a brown crossbody bag with a strap across her shoulder. The bag appears to be made of leather and has a small flap closure. The overall style of the outfit is casual and minimalistic."
- "A close-up portrait of a young woman's face and upper body. She is wearing a black strapless top with a thin silver chain necklace around her neck. Her hair is styled in loose waves and she is wearing large hoop earrings. The woman is looking off to the side with a serious expression on her face. The background is plain white."
- "A close-up portrait of a woman's upper body. She is wearing a black collared shirt with a button-down collar and long sleeves. Her hair is styled in loose curls and she is wearing large, dangling earrings. Her hand is resting on her chest, with a large ring on her ring finger. The background is plain white. The woman appears to be looking off to the side with a serious expression on her face."

""

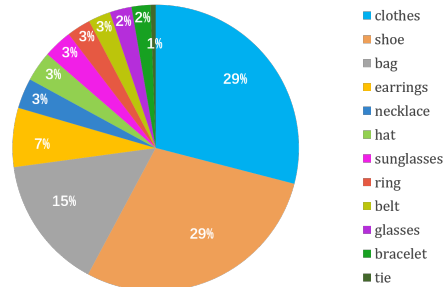
B Details of Training Dataset

B.1 Dataset for Stage-1

The model in the first stage is jointly trained on two datasets, *i.e.*, the unpaired in-the-wild images, and the dataset of stage-2 without the object image. We train on the datasets with sampling ratios of 2 : 1. To further investigate the class distribution in the unpaired dataset, we count the highly-frequent words in the object text descriptions. After filtering out the prepositions and verbs, the top-5 words are necklace, hat, glasses, sunglasses and watch. We also observe some classes excluded in our final 12 common classes, *e.g.*, smartphone, cup, scarf, crown and mask. The rich distribution of wearable or holdable objects enhances the generalization of OmniTry to uncommon classes.

We also report the scale of dataset during the data preparation. The initial dataset contains 152K in-the-wild images, which are filtered to be 111K images with person and wearable objects. After listing, grounding and removing objects, the total amount of images containing at least one object is 94K, and the corresponding number of objects is 189K (roughly 2 objects per image).

Figure 8: The class distribution of training dataset.



B.2 Dataset for Stage-2

For the training dataset of the second stage, we visualize the amount of samples for each class in Appendix B. It is shown that the most common classes, *i.e.*, clothes and shoes, constitute more half of the total dataset, while most classes lay in the long-tail of distribution with less than 3%. Such a distribution is aligned with our basic assumption that it is hard to obtained paired samples for many wearable objects. For class-balanced training, we manually assign the sampling weights for clothes, shoes and bags as 4, 4, 3, and set weights as 1 for remaining classes.

C Details of Training and Model Architecture

C.1 Training Configuration

During training, we resize the image with fixed aspect ratio to be no larger than 1 million, which means that the model could receive images with varying aspect ratios in one batch. To handle this, we pad the image tokens into the same length of sequence, and modify the attention block to forward only on the valid tokens.

For both training of stage-1 and stage-2, we set the learning rate as 1^{-4} , gradient accumulation steps as 1, weight decay as 0.01 and gradient norm clipping as 1.0. We use the AdamW [38] optimizer with hyper-parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The model is trained with mixed precision of bfloat16. We note that since we fine-tune based on the distilled version of FLUX [32], the guidance scale is fixed as 1 during training, and set as 30 during inference.

C.2 Details of Re-purposing Inpainting Model

We elaborate the details of adapting the inpainting model, FLUX.1-Fill in this paper, towards mask-free try-on task. During training, the input of model can be split into two sets in sequence dimension:

- The try-on image. Along the channel dimension, it contains the noisy ground-truth try-on image, the input person image and a zero mask in the same shape.
- The object image. Along the channel dimension, it contains the noisy object image, the clean noisy image and a zero mask.

Then during the inference stage, we initialize the above input while replacing the noisy latents with standard Gaussian noise. Through the above formulation, it is shown that the inputs of person and object images are different. The person branch aims to modify the input person image in proper area, while the object branch simply targets to maintain the input, and transfers the object appearance via full attention mechanism.

C.3 Details of Masked Full-Attention

We discuss the details of applying masked full-attention in the second stage. We set text prompts for both try-on and object images, like “trying on sunglasses”. Suppose the length of tokens to be: L_{I1} for try-on image, L_{T1} for try-on text, L_{I2} for object image, and L_{T2} for object text. We concatenate all tokens in the above order. Then the attention mask is:

$$\begin{bmatrix} 1_{L_{I1} \times L_{I1}} & 1_{L_{I1} \times L_{T1}} & 1_{L_{I1} \times L_{I2}} & 0_{L_{I1} \times L_{I1}} \\ 1_{L_{I1} \times L_{I1}} & 1_{L_{I1} \times L_{T1}} & 0_{L_{I1} \times L_{I2}} & 0_{L_{I1} \times L_{I1}} \\ 0_{L_{I1} \times L_{I1}} & 0_{L_{I1} \times L_{T1}} & 1_{L_{I1} \times L_{I2}} & 1_{L_{I1} \times L_{I1}} \\ 0_{L_{I1} \times L_{I1}} & 0_{L_{I1} \times L_{T1}} & 1_{L_{I1} \times L_{I2}} & 1_{L_{I1} \times L_{I1}} \end{bmatrix}, \quad (2)$$

where $1_{m \times n}$ denotes all-one matrix and $0_{m \times n}$ denotes all-zero matrix. More specifically, we apply such a full-attention in both the multi-modality blocks and single blocks of FLUX [32], and figure out the text tokens to achieve the masking. We leverage the attention function with varying length in FlashAttention [12] to implement the block-wise masked attention.

C.4 LoRA Implementation

We implement the location and identity adapters with LoRA [21]. In detail, we set the rank and α to be 16. We insert the LoRA module into the following layers: the projection into query/key/value, output projection of attention, the linear layers in feedforward block, the layer normalization layer, the input patch projection, and the final output projection.

D Details of Compared Methods

In this section, we present the details of compared methods and our implementation of them on try-on task. We also report more results of the variants of each method, among which we only report the best result in main experiment.

D.1 General Customized Image Generation

OneDiffusion [33]: A large-scale diffusion framework supporting bidirectional image synthesis across tasks. We evaluated its performance on mask-free/mask-based try-on through instruction-based cases. We also modify its original instructing prompt to achieve better performance.

OmniGen [58]: A vision-language unified framework consolidating multiple tasks, supporting both mask-free/mask-based generation. We also test it with both standard and our optimized prompts.

VisualCloze [35] implements visual in-context learning for domain generalization. We conduct experiments with single example and multiple examples in the context.

Paint-by-Example [61] enables to re-paint a given subject into image via CLIP-based object representation with mask dependency.

MimicBrush [5] achieves imitative inpainting for region-specific edits, requiring the input image with mask, together with the reference image without mask.

ACE++ [40] extends long-context conditioning for instruction-driven generation that tackles various.

D.2 Image-based Virtual Try-On

OOTDiffusion [60] designs a two-branch U-Net architecture to consume the person and garment images, which requires masked input in the person branch.

Magic Clothing [4] introduces a garment extractor to progressively insert garment features into the main backbone of try-on generation. Magic Clothing supports the input of either masked person image, or the targeting pose and person ID image. We adapt the former setting to better preserve the person image.FI

CatVTON [10] proposes to transfer the identity of garment by simply concatenating it with the person image, and achieve mask-based try-on with inpainting model.

FitDiT [25] introduces diffusion transformer (DiT) model into VTON, and designs a GarmentDiT and a DenoisingDiT to implement this task.

Any2AnyTryon [18] is the only open-source mask-free VTON model, eliminates the dependence on masks, poses, or any other such conditions.

D.3 More Comparison Results

We report more comparison results in Tab. 3, including variants of methods with mask/mask-free setting, varying image size and different prompt design. We report only the best result of all variants in the main experiment.

E More Visualization Results

We visualize more try-on results in Fig. 9, where we include all classes in OmniTry-Bench and different sub-types for full visualization.

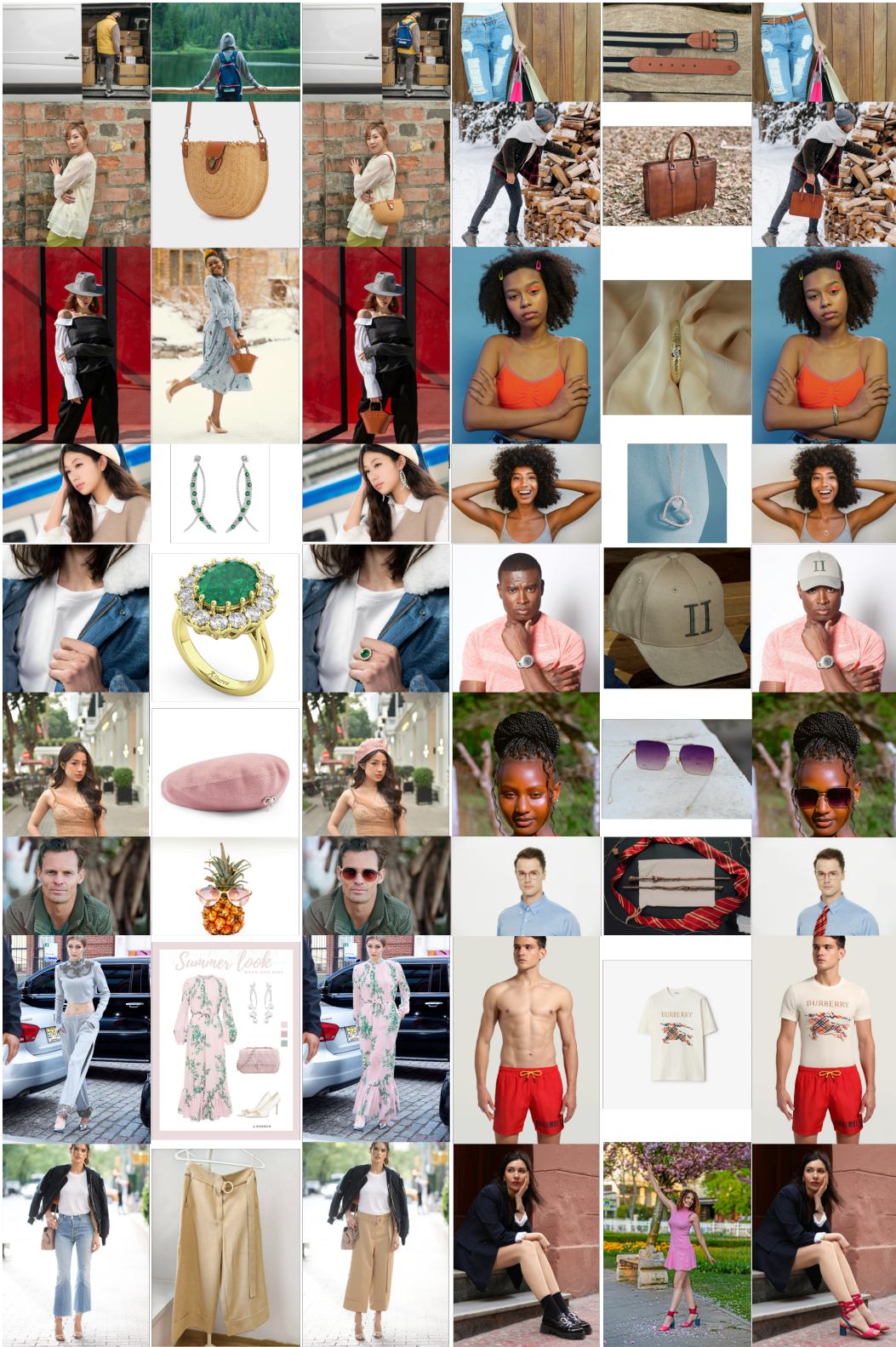


Figure 9: The samples of the model, the object, and the try-on person.

Table 3: More evaluation results of the compared methods with different settings.

method	mask	Object Consistency		Person Presevation		Object Localization	
		M-DINO \uparrow	M-CLIP-I \uparrow	LPIPS \downarrow	SSIM \uparrow	G-Acc. \uparrow	CLIP-T \uparrow
on the whole set							
Paint-by-Example (512 ²) [61]		0.4171	0.7328	0.4577	0.7968	0.9833	0.2831
Paint-by-Example (1024 ²) [61]		0.4565	0.7727	0.3903	0.8033	0.9861	0.2804
MimicBrush [5]		0.4693	0.7253	0.3033	0.8575	0.9250	0.2781
ACE++ (prompt v1) [40]		0.4565	0.7474	0.4561	0.7519	0.9667	0.2791
ACE++ (prompt v2) [40]	✓	0.4449	0.7427	0.4554	0.7517	0.9722	0.2793
VisualCloze (1-example) [35]		0.4705	0.7533	0.6685	0.5320	0.9972	0.2283
VisualCloze (2-example) [35]		0.4236	0.7307	0.6767	0.4908	0.9917	0.2260
OmniGen (prompt v2) [58]		0.5151	0.7761	0.6888	0.5870	0.9917	0.2557
OneDiffusion (prompt v1) [33]		0.5515	0.8137	0.6607	0.6166	1.0	0.2290
OneDiffusion (prompt v2) [33]		0.5580	0.7950	0.5795	0.6628	0.9972	0.2401
OneDiffusion (prompt v1) [33]		0.4178	0.7358	0.7606	0.4951	1.0	0.2309
OneDiffusion (prompt v2) [33]		0.4731	0.7749	0.7001	0.5831	0.9972	0.2309
VisualCloze (1-example) [35]		0.5292	0.7782	0.4471	0.6190	0.9639	0.2524
VisualCloze (2-example) [35]	✗	0.4915	0.7619	0.4730	0.5868	0.9806	0.2540
OmniGen (prompt v1) [58]		0.5299	0.7689	0.7009	0.5727	0.9778	0.2533
OmniGen (prompt v2) [58]		0.5435	0.7869	0.6703	0.5965	0.9944	0.2535
OmniTry (Ours)		0.6160	0.8327	0.0542	0.9333	0.9972	0.2831
on the clothes subset							
Magic Clothing [4]		0.5665	0.7634	0.2761	0.8786	1.0	0.2700
CatVTON [10]		0.5744	0.7906	0.1664	0.9283	1.0	0.2818
CatVTON (w. garment mask) [10]		0.5534	0.7843	0.2084	0.8828	1.0	0.2797
OOTDiffusion [60]	✓	0.5961	0.8016	0.2178	0.8865	1.0	0.2761
FitDiT (768 \times 1024) [25]		0.6718	0.8324	0.1972	0.8952	1.0	0.2822
FitDiT (1152 \times 1536) [25]		0.6733	0.8340	0.1618	0.9027	1.0	0.2831
FitDiT (1536 \times 2048) [25]		0.5961	0.8016	0.2178	0.8865	1.0	0.2761
Any2AnyTryon [18]	✗	0.6747	0.8537	0.2089	0.8969	1.0	0.2832
OmniTry (Ours)		0.6995	0.8560	0.1021	0.9105	1.0	0.2799

Table 4: Human evaluation results of OmniTry and garment-only methods.

Method	Magic Clothing	CatVTON	OOTDiffusion	FitDiT	Any2AnyTryon	OmniTry (Ours)
Avg. Rank \downarrow	4.27	3.36	3.70	2.28	0.77	0.62

F Human Evaluation of the Generated Try-ons

We conduct a human evaluation to assess the realism and usefulness of the generated try-on results, especially in comparison with garment-only methods. Specifically, we invite five annotators to rank the outputs of different methods based on three aspects: try-on success rate, garment consistency, and overall realism. The average ranking results are summarized in Tab. 4, where a lower value indicates a better ranking. As shown, OmniTry achieves the best overall performance among all compared methods.

G Differences between Stage-1 of OmniTry and Editing Methods

The key differences between the stage-1 of OmniTry and the editing methods that support the ‘‘Add’’ operation can be summarized as follows. (1) Task and performance: The general editing methods typically involve a wide range of editing tasks, thus may show restricted performance on specific operation, especially on try-on cases requiring fine-grained combination of the added object and the original image. The added object could be more likely to be an independent item, while OmniTry focuses on natural combination with parts of input person. (2) Method: The stage-1 of OmniTry is designed by re-purposing an inpainting-based model to mask-free editing, leveraging its ability of

Table 5: Comparison between OmniTry (stage-1) and editing methods supporting “Add” operation.

Method	LPIPS	SSIM	G-Acc.	CLIP-T
AnyEdit	0.1112	0.8455	0.8167	0.2415
OmniGen	0.3381	0.6394	0.9889	0.2654
OmniTry (stage-1)	0.0711	0.8959	0.9944	0.2613

Table 6: Additional ablation study on one-stream vs. two-stream adapter.

Method	trainable params.	M-DINO	M-CLIP	LPIPS	SSIM	G-Acc.	CLIP-T
two-stream	172M (2 LoRA with r=16)	0.5845	0.8159	0.0425	0.9403	0.9806	0.2620
one-stream	172M (1 LoRA with r=32)	0.5619	0.8149	0.0439	0.9478	0.9861	0.2604

detailed local editing. Specifically, the original image and generated image are concatenated in the channel dimension. However, the general editing methods require larger divergence between the input and output, and are concatenated in the sequential dimension (e.g., UniReal [7] and OmniGen [58]), showing higher computation cost (2x sequence length).

We compare AnyEdit [26] and OmniGen [58] with the stage-one model of OmniTry in Tab. 5, with the metrics of object localization and person preservation. We observe that OmniGen could not guarantee to preserve the original image (similar to its performance in stage-2). For AnyEdit, though it preserve the input image, it could sometimes fail to add any object (worse G-Acc.) or properly combine the object onto the person. We will also include visualization result in revised version.

H Additional Ablation Study on One-Stream vs. Two-Stream Adapter

To further ensure the alignment of trainable parameters, we train a new location adapter with double LoRA rank (r=32) from stage-1, and initialize it into the second stage for one-stream training. We note that the additional computation cost is doubled than two-stream adapters with r=16. We initialize both settings from an earlier checkpoints of stage-1 with the same training steps to ensure fair comparison. The results in Tab. 6 show that though with less computation cost, the two-stream setting still shows better performance to separately cope with different capabilities of OmniTry.

I More Discussion on Unexpected Shortcut in Stage-1

In stage-1, we observe that using naively erased training samples leads the model to produce output images that almost perfectly recover the position and shape of the object in the ground-truth image. We hypothesize that this phenomenon is likely caused by information leakage, for the following reasons. (1) The reconstruction shown in Fig. 3 primarily reflects shape and position reconstruction, rather than appearance reconstruction. Since no object image is provided to the first stage, the model generates objects with diverse appearances but consistently reproduces the same shape and position as the ground-truth object. This observation suggests that the model might exploit the boundary of the erased region, enabling it to perfectly reconstruct the object’s location and outline. (2) A stronger piece of evidence is observed when we train the model with traceless erasing under the same number of training steps. In this case, the model produces objects with random shapes, positions, and appearances, even when evaluated on the training samples, indicating that the shape recovery in the naive erasing setup indeed stems from boundary leakage.