DiffUHaul: A Training-Free Method for Object Dragging in Images

OMRI AVRAHAMI, NVIDIA, The Hebrew University of Jerusalem
RINON GAL, NVIDIA, Tel Aviv University
GAL CHECHIK, NVIDIA
OHAD FRIED, Reichman University
DANI LISCHINSKI, The Hebrew University of Jerusalem
ARASH VAHDAT*, NVIDIA
WEILI NIE*, NVIDIA

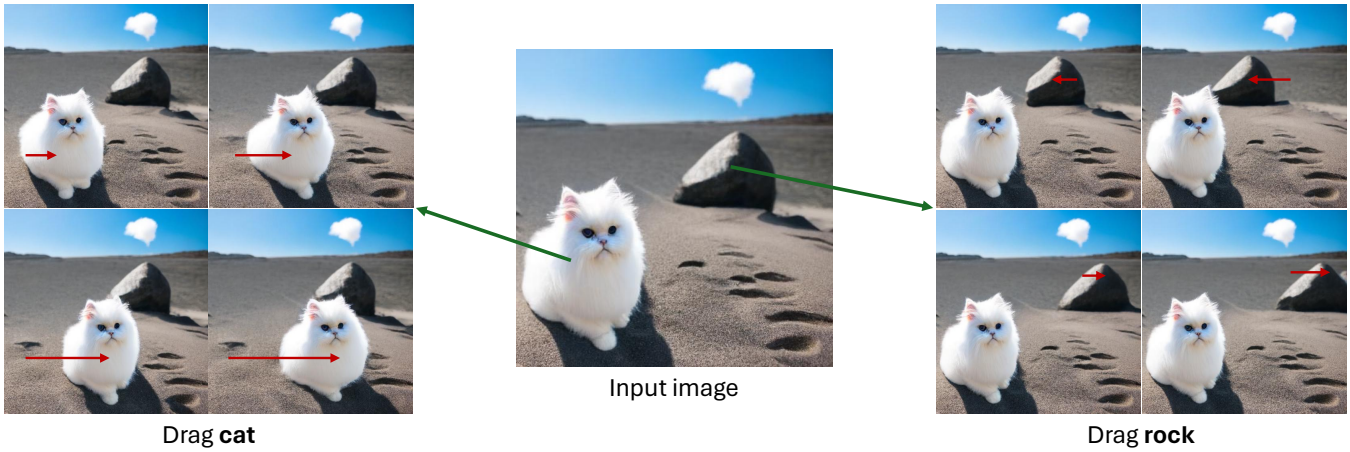

Fig. 1. **DiffUHaul:** Given a real image with multiple objects (e.g., a cat and a rock), our method is able to seamlessly drag each of the objects to an arbitrary location within the image while preserving the foreground and background appearance.

Text-to-image diffusion models have proven effective for solving many image editing tasks. However, the seemingly straightforward task of seamlessly relocating objects within a scene remains surprisingly challenging. Existing methods addressing this problem often struggle to function reliably in real-world scenarios due to lacking spatial reasoning. In this work, we propose a training-free method, dubbed *DiffUHaul*, that harnesses the spatial understanding of a *localized* text-to-image model, for the object dragging task. Blindly manipulating layout inputs of the localized model tends to cause low editing performance due to the intrinsic entanglement of object representation in the model. To this end, we first apply attention masking in each denoising step to make the generation more disentangled across different objects and adopt the self-attention sharing mechanism to preserve the high-level object appearance. Furthermore, we propose a new diffusion anchoring technique: in the early denoising steps, we interpolate the attention features between source and target images to smoothly fuse new layouts with the original appearance; in the later denoising steps, we pass the localized features from the source images to the interpolated images to retain fine-grained object details. To adapt DiffUHaul to real-image editing, we apply a DDPM self-attention bucketing that can better reconstruct real images with the localized model. Finally, we introduce an automated evaluation pipeline for this task and showcase the efficacy of our method. Our results are reinforced through a user preference study.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Computer graphics**.

Additional Key Words and Phrases: Object Draggining, Image Editing

*Indicates Equal Advising

1 INTRODUCTION

Think about a digital artist who recently employed an advanced generative model to craft an image featuring a Persian cat alongside a rock, as in Figure 1. All that is needed for their creation to achieve perfection is for the cat (or the rock) to be moved slightly. Despite the conceptual simplicity of such a task, seamlessly dragging objects in an image is surprisingly challenging for current generative image editing methods [Brooks et al. 2023; Hertz et al. 2022]. In this work, we propose a novel training-free solution for this scenario.

Current methods that tackle this problem rely on time-consuming LoRA training per image [Shi et al. 2023], training a designated model on a large dataset [Chen et al. 2023a; Yang et al. 2022] or utilizing classifier-free guidance (CFG) with specific objectives [Epstein et al. 2023; Mou et al. 2023, 2024]. However, these methods are not robust and struggle to operate reliably in a real-world setting. For example, as can be seen in Figure 2, DiffEdit [Mou et al. 2024] suffers from artifacts of traces of the puppy in its original location, while our method demonstrates a more robust behavior.

Recently, several *localized* text-to-image models were developed by the community that add spatial controllability to the task of text-to-image generation [Avrahami et al. 2023c; Li et al. 2023; Nie

Project page is available at: https://omriavrahami.com/diffuhaul/

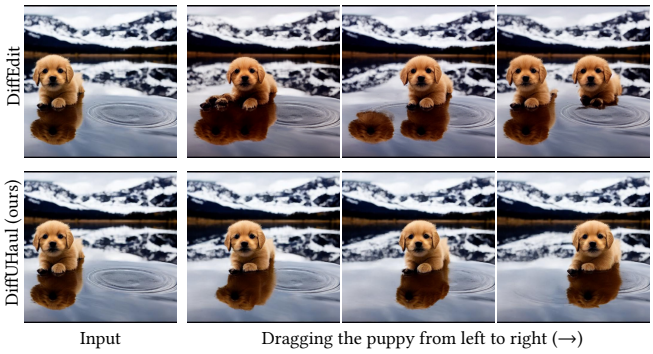

Fig. 2. **Object Dragging Robustness.** When dragging a puppy in a complex environment (particularly with its reflection in the water and ripples nearby) to different locations along from left to right, previous method DiffEdit [Mou et al. 2024] struggles with the editing traces left in its original location, while our method demonstrates a more robust behavior.

et al. 2024; Yang et al. 2023; Zhang et al. 2023; Zheng et al. 2023]. A natural question is then whether the localized understanding of the 2D pixel world in such models can be harnessed for the task of object dragging. Hence, we examine the disentanglement properties of such models, and propose a series of modifications that allow them to serve as a backbone for drag-and-drop movement of objects within an image. Specifically, we use the recently introduced BlobGEN [Nie et al. 2024] model, and demonstrate that its spatial understanding can enable significantly more robust object dragging without requiring fine-tuning or training.

In pursuit of our solution, we begin by revealing an entanglement problem in the localized text-to-image models, through which the prompt-based localized controls of different image regions interfere with each other. We trace the root cause to the commonly used Gated Self-Attention layers [Li et al. 2023], where each individual layout embedding are free to attend to all the visual features. We propose an inference-time masking-based solution, named *gated self-attention masking*, and show that improving the model disentanglement leads to better object dragging performance.

Next, specially for the object dragging task, we first adopt the commonly-used self-attention sharing mechanism [Cao et al. 2023] to preserve the high-level object appearance. To better transfer the fine-grained object details from source images to target images and better harness spatial understanding of the model, we propose a novel soft anchoring mechanism: in early denoising steps, which control the object shape and scene layout in an image, we interpolate the self-attention features of the source image and those of the target image with a coefficient relative to the diffusion time step. This process promotes a smooth fusion between the target layout and source appearance. Then, in later denoising steps, which control the fine-grained visual appearance in an image, we update the interpolated attention features from the corresponding features in the source image via the nearest-neighbor copying.

To adapt our method to real-image editing, we further require an inversion solution that is compatible with the localized method. We find that the standard DDIM inversion [Song et al. 2020] struggles to reconstruct the image faithfully, even when not using classifier-free guidance [Ho 2022]. Hence, we propose a simple DDPM self-attention bucketing technique that adds noise to the reference image *independently* in each diffusion step, and uses the noisy images to extract the self-attention outputs as the source attention features. This DDPM bucketing does not accumulate reconstruction errors along the denoising process and preserves details for real images.

Finally, we offer automatic metrics for our problem to assess different aspects of the editing operations, and use them for an extensive comparison that demonstrates the effectiveness of our method over the baselines. In addition, we conduct a user study and show that our method is also preferred by human evaluators.

In summary, our contributions are: (1) we show that the spatial understanding of a localized text-to-image model can be effectively harnessed to tackle the object dragging task, (2) we reveal an entanglement problem in the gated self-attention layers and offer an inference-time solution, (3) we introduce a novel soft anchoring mechanism that fuses the source object appearances and the target scene layouts during the denoising process, (4) we show that DDPM self-attention bucketing suffices for real image editing, and finally (5) we develop automatic metrics to the task of object dragging and use them to evaluate our method quantitatively, in addition to a user study, to demonstrate its effectiveness.

2 RELATED WORK

Localized text-to-image models. Recently, text-to-image diffusion models [Ho et al. 2020; Ramesh et al. 2022; Rombach et al. 2021; Sohl-Dickstein et al. 2015; Song et al. 2020; Song and Ermon 2019; Yu et al. 2022] became a foundational tool for creative tasks [Avrahami et al. 2023d; Frenkel et al. 2024; Molad et al. 2023; Richardson et al. 2023]. To add spatial control to existing text-to-image models, some works suggested training a designated localization component to take in visual layouts [Avrahami et al. 2023c; Li et al. 2023; Nie et al. 2024; Yang et al. 2023; Zhang et al. 2023; Zheng et al. 2023] while others offer training-free methods that incorporate the spatial conditioning into the diffusion sampling process [Bar-Tal et al. 2023; Chefer et al. 2023; Chen et al. 2023b; Feng et al. 2022; Phung et al. 2023]. In this work, we utilize BlobGEN [Nie et al. 2024] as our base model since it has shown better spatial understanding and generation quality.

Text-to-image editing. Soon after the emergence of text-to-image diffusion models, a plethora of methods were offered for various image editing tasks [Avrahami et al. 2023b, 2022; Cao et al. 2023; Hertz et al. 2023; Kawar et al. 2023; Meng et al. 2021; Mokady et al. 2023; Patashnik et al. 2023; Sheynin et al. 2023; Tumanyan et al. 2023]. However, most of these editing methods are spatially preserving (i.e., changing the object attributes and categories), and suffer from the editing tasks requiring spatial reasoning, such as object dragging [Brooks et al. 2023; Hertz et al. 2022]. Localized text-to-image models, such as GLIGEN [Li et al. 2023] and BlobGEN [Nie et al. 2024], has the potential to solve the object dragging task, but their performance is far from satisfactory without specialized designs.

Keypoint dragging. A similar task is keypoint dragging, where users provide source and target keypoints in the image, and move the source keypoints to the target ones. For example, UserControllableLT [Endo 2022], GANWarping [Wang et al. 2022] and DragGAN [Pan et al. 2023] employ StyleGAN [Karras et al. 2021, 2019, 2020] for editing generated images. But they work only on the narrow domain the GAN [Goodfellow et al. 2014] was trained on (e.g., human faces, churches [Yu et al. 2015]). DragDiffusion [Shi et al. 2023] propose a LoRA-based [Hu et al. 2021] method that finetunes a diffusion model given a test image and optimizes the latent noises at inference time. In contrast, our method is training-free.

Object dragging. Different from keypoints dragging that warps the image to match the target keypoints, object dragging moves the entire object seamlessly to a new position. Object dragging was initially introduced by [Epstein et al. 2022; Wang et al. 2021] for single-domain images generated by GANs. Diffusion self-guidance [Epstein et al. 2023] proposed to use the guidance from internal representations of a diffusion model for various editing tasks, including object dragging. DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] developed a new classifier guidance [Dhariwal and Nichol 2021] specifically designed for object dragging. Most of them use a general diffusion model as the base model, but our method harnesses the spatial understanding of a *localized* diffusion model to better tackle the object dragging task.

Object insertion. Many works use multiple images [Alaluf et al. 2023; Arar et al. 2024; Gal et al. 2022; Ruiz et al. 2023; Voynov et al. 2023] or a single image [Arar et al. 2023; Avrahami et al. 2023a; Gal et al. 2023] of the same object for image personalization. They are also effective in tackling the task of referenced-based object insertion, in which a reference object is being inserted to a target image. AnyDoor [Chen et al. 2023a] and PaintByExample [Yang et al. 2022] train a designated encoder for this task, which can be used for object dragging by utilizing an inpainting method, as explained in Section 5. The concurrent work ObjectDrop [Winter et al. 2024] collected a high-quality tailored dataset to train a model for object removal, insertion, and dragging. Our method, however, is training-free with a pre-trained *localized* diffusion model.

3 PRELIMINARIES

Existing large text-to-image diffusion models suffer from the prompt-following issue, making it challenging to control the visual layouts of their generation via complex prompts only. Thus, incorporating the visual layout information into these large text-to-image diffusion models can enable better object-level controllability [Avrahami et al. 2023a; Li et al. 2023; Yang et al. 2023]. Among them, visual layouts are usually represented by bounding boxes (along with object categories).

More recently, BlobGEN [Nie et al. 2024] has introduced a new type of visual layouts called blob representations to guide the image synthesis, which shows more fine-grained controllability than all previous approaches. Specifically, the blob representations denote the object-level visual primitives in a scene, each of which consists of two components: blob parameters τ and blob description S. A blob parameter depicts a tilted ellipse using a vector of five variables $\tau = [c_x, c_y, a, b, \theta]$ to specify the object's position, size and orientation, where (c_x, c_y) is the center point of the ellipse, a and b are the radii of its semi-major and semi-minor axes, and $\theta \in (-\pi, \pi]$ is the orientation angle of the ellipse. A blob description S captures the object's visual appearance using a region-level synthetic caption extracted by an image captioning model. Compared with bounding boxes and object categories, the blob representations can retain more detailed spatial and appearance information about the objects in a complex scene.

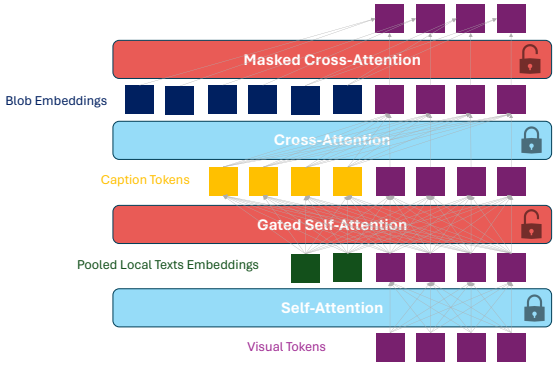

Fig. 3. **BlobGEN Architecture.** BlobGEN incorporates the additional blob information into the Stable Diffusion model by adding two new layers in each attention block: masked cross-attention and gated cross-attention.

To incorporate blob representations into the existing Stable Diffusion model, BlobGEN adopts a similar architecture design idea to GLIGEN [Li et al. 2023] that introduces new attention layers in a gated way. To retain the prior knowledge of pre-trained models for synthesizing high-quality images, it freezes the weights of the pre-trained diffusion model and only trains the newly added layers. As demonstrated in Figure 3, BlobGEN keeps the gated self-attention module originally developed by GLIGEN while also introducing a new masked cross-attention module in each attention block. These two new layers fuse blob inputs into the model differently: In the gated self-attention layer, the blob embeddings are first passed to a pooling layer and then concatenated with the visual features, while, in the masked cross-attention layer, each blob embedding only attends to visual features in its local region as the feature maps are masked by the (rescaled) blob ellipses.

With this masking design, each blob representation and its local visual feature are trained to align with each other, and thus the model becomes more modular and disentangled. BlobGEN has demonstrated more fine-grained control over its generation. Therefore, we use BlobGEN as our network backbone for solving the object dragging task.

4 METHOD

Our goal is to offer a solution to the problem of object dragging. To this end, we propose to leverage the spatial knowledge of blob-based text-to-image model BlobGEN [Nie et al. 2024]. In Section 4.1 we start by investigating the disentanglement offered by this model. We discover significant lingering entanglement, and trace it to the gated-self attention of GLIGEN-style models. Hence, we offer an

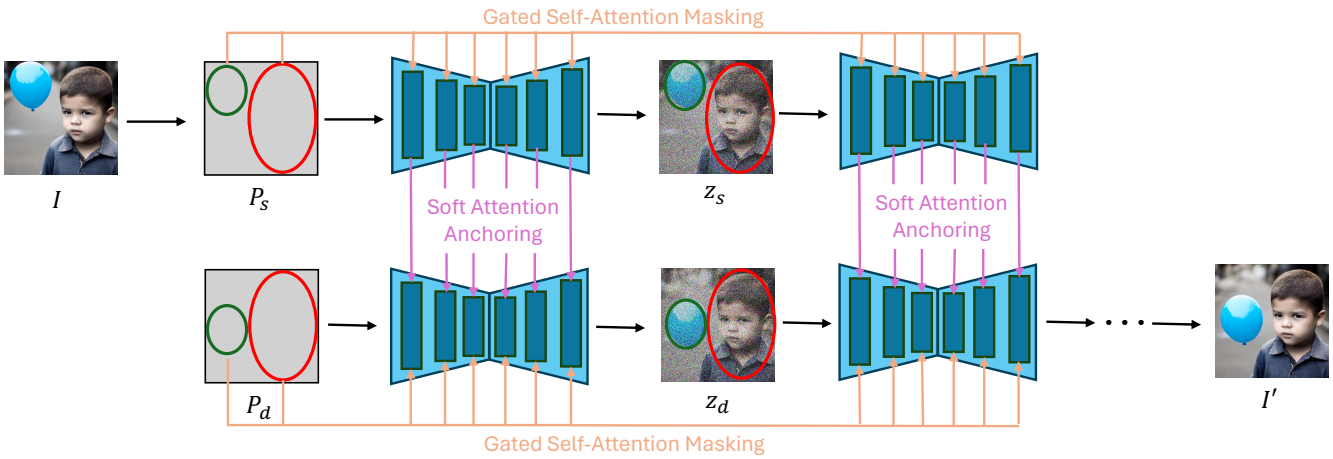

Fig. 4. **Method Overview.** Given an input image I, we start by extracting the blob parameters P_s of its layout; then, by changing its layout based on the user provided target location, we get the new blob parameters P_d. By conditioning the localized text-to-image model on the respective blob representations, we iteratively denoise the source and target images (z_s and z_d) while incorporating gated self-attention masking (Section 4.1) and soft attention anchoring (Section 4.2) in each self-attention block until we get the desired editing result I'.

inference-time mask-based solution to this problem. In Section 4.2 we present our solution for object dragging in generated images: (1) we first utilize self-attention sharing [Cao et al. 2023; Geyer et al. 2023; Tewel et al. 2024; Wu et al. 2022] to increase the consistency of the dragged object, and (2) we propose a soft anchoring technique to improve the consistency of results. Finally, in Section 4.3 we extend our solution to real images by relying on the proposed DDPM self-attention bucketing instead of standard DDIM inversion. Our method is summarized in Figure 4.

Formally, given an input image I with an object located in (c_x, c_y) that the user wants to drag, and a desired target location (c'_x, c'_y), the task of object dragging aims at moving the object to the target location while the rest of the image is left intact, up to desired environment changes (e.g., reflections) in the edited image I'.

4.1 Gated Self-Attention Entanglement

As explained in Section 3, BlobGEN was trained to take a set of input blobs $B_1, ...B_n$ with corresponding text descriptions $S_1, ...S_n$ and blob parameters $\tau_1, ...\tau_n$, and generate a scene. This scene is expected to be created in a *disentangled* manner, i.e., the text description S_i should correspond only to the local region depicted by τ_i. To this end, the authors introduced a masked cross-attention layer. However, a simple investigation reveals that the generated result is not fully disentangled in practice. For example, as can be seen in Figure 5 (first row), the rabbit text description from one blob spills over to the spatial region of the cat blob.

We hypothesize that the gated self-attention modules that BlobGEN derives from GLIGEN is the root cause of entanglement. In gated self-attention, a projection layer first converts the CLIP [Radford et al. 2021] text embeddings of the text description S_i to the text tokens $T = \{t_1, ...t_n\}$. They are then merged with the visual tokens $V = \{v_1, ...v_k\}$ into a unified set $V \cup T = \{v_1, ...v_k, t_1, ...t_n\}$, which altogether are used to calculate the self-attention features, using the standard self-attention mechanism (plus a gated skip connection).

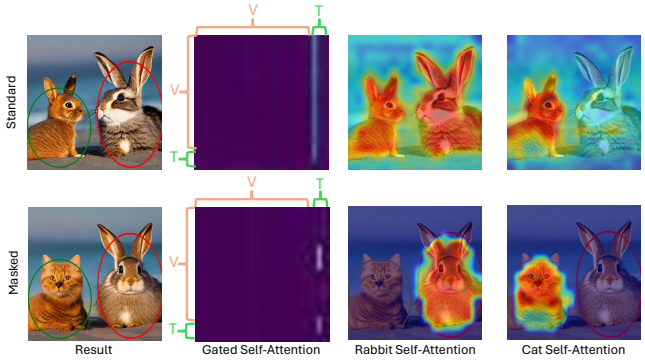

Fig. 5. **Gated Self-Attention Leakage.** Given scene descriptions of two blobs: *"a photo of a rabbit"* and *"a photo of a cat"*, the standard BlobGEN model (first row) leaks the rabbit information also to the cat blob (the first row third column), while our masked version of the gated self-attention (second row) is able to disentangle the blobs (the second row third column). In addition, we can see that the gated self-attention (second column) behaves de facto as a cross-attention layer, as the vast majority of the attention is between the text tokens T and the visual tokens V.

This design choice adds no constraint over the attended areas, i.e., the projected text tokens T can attend to themselves and all the visual tokens V. To visualize this phenomenon, we average the gated self-attention maps over the diffusion process. An example is shown in Figure 5 (the second column of the first row). This visualization reveals an interesting aspect: the vast majority of attention weights is between the projected text tokens T and the visual tokens V, and not within these sets themselves. It means that the gated self-attention layer behaves as a de facto cross-attention layer.

We examine the attention between the projected text token t_i and all the visual tokens V. This is a K-dimensional vector, which we first reshape into two dimensions $\sqrt{K} \times \sqrt{K}$, and then resize

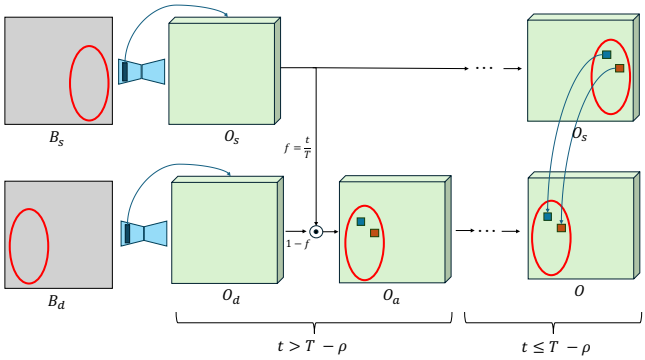

Fig. 6. **Self-Attention Soft Anchoring.** Given the source blob B_s and target blob B_d, we start by extracting the self-attention outputs O_s and O_d correspondingly, then, during the first ρ iterations, we blend these maps according to the timestep ratio $f = \frac{t}{T}$ where t is the current timestep and T is the total number of timesteps. Then, after the anchor map O_a is calculated, we use it for determining the position of the new blob, while taking the appearance from the corresponding O_s map using nearest-neighbor copying.

to a canonical size. We term these maps "reshaped self-attention", which are averaged over all the denoising steps. This visualization, as shown in Figure 5 (last two columns of the first row), reveals that text tokens indeed attend to undesired areas: the "rabbit" text token attends to the visual features in both the "rabbit" and "cat" blob regions, leading to an *entangled* generation. For more details about the visualizations, please refer to the supplementary material.

To this end, we suggest an inference-time solution to the entanglement problem: given n different input blobs with the corresponding parameters $\tau_1, ...\tau_n$ we first convert them into n masks $M_1, ...M_n$ of 512×512 resolution. Then, during the diffusion process, for each self-attention layer and for each projected text token t_i, we reshape the mask M_i to the corresponding spatial size of the layer, and use it to mask the area of the gated self-attention between the projected text token t_i and the visual tokens V. This way, we can prevent the token t_i from attending to undesired areas at the inference time.

4.2 Consistent Object Dragging for Generated Images

Now, we first focus on tackling the object dragging problem for generated images from the localized model: given a scene represented by n blob inputs $B_1, ...B_n$, we change the parameters τ_s of one blob B_s to τ_d with a different spatial location such that the s^{th} object in the generated image will be relocated to the designated location, without changing the appearance of all other objects and the background (barring direct interactions with the object, e.g., shadows).

To preserve the high-level object appearance, we adopt the self-attention sharing mechanism [Cao et al. 2023; Wu et al. 2022]: we iteratively generate the source image using the source parameters τ_s in parallel to the target image with the τ_d parameters. Then, we replace the self-attention keys K_d and values V_d from the target image in each self-attention layer and each denoising step by the keys and values K_s, V_s from the source image.

However, this mechanism alone does not fully preserve the fine-grained details of the source image, so we propose adding a novel *soft anchoring mechanism*: the motivation is that the generated source image already contains the information needed for generating the target image, we can take advantage of the self-attention layers *output* (i.e., attention features) in the local region that corresponds to the source blob. The soft anchoring is designed to fuse the object appearance information represented by the attention features within the source blob and the positional information indicated by the target blob. Specifically, in the first ρ steps of the denoising process, we perform an adaptive, *soft blending* of the attention features of the generated target image with the features of the source image. The interpolation coefficient is time-dependent: we take more visual appearance from the source image in the beginning but more spatial information from the target image in the later steps, as depicted in Figure 6. Formally, for each denoising step $t \in [T, T-1, ..., T-\rho+1]$ and for each self-attention layer, the interpolated self-attention output of the target image is:

$$O_a = O_s * f + O_d * (1 - f); f = \frac{t}{T}$$

where O_s is the self-attention output of the generated source image, O_d is the self-attention output of the generated target image, and T is the total number of denoising steps. The length of soft blending is controlled by the hyperparameter ρ.

Next, during the last $T - \rho$ steps of the denoising process, we use the soft blending result O_a as anchor points for the target object. In each denoising step $t \in [T - \rho, ..., 2, 1]$ and each self-attention layer, we perform the *nearest-neighbor copying*: each entry from the anchor attention features O_a within the target blob B_d is replaced by its nearest-neighbor entry from the source attention features O_s within the source blob B_s. The nearest-neighbor entry is obtained by measuring the normalized cosine similarity. Formally,

$$(O_a)_{(j,k)\in B_d} = (O_s)_{NN(j,k)\in B_s}$$

where $(j, k) \in B_d$ represents the set of coordinates for each entry from O_a within the target blob B_d and $NN(j, k) \in B_s$ denotes the set of coordinates for each nearest-neighbor entry from O_s within the source blob B_s.

4.3 Extension for Real Images

In order to extend our method for dragging objects in real images, we first extract the blobs parameters as we explain later, then, we need to invert the image. However, we found that directly applying the DDIM inversion [Song et al. 2020] in a localized model is not able to preserve the details of the input image, even without classifier-free guidance [Ho 2022].

Recall that when dealing with generated images, the input signal from the source image is fed in our pipeline through its self-attention outputs. Hence, we only need to extract the self-attention features in different attention layers and different denoising steps from the real image, rather than an actual inversion that searches for the optimal latent noises. To this end, we propose the *DDPM self-attention bucketing*: we first add *independent* noises with various scales to the real image, where the noise scale corresponds to a time step in the DDPM forward process. The noisy images at every time step,

Table 1. **Quantitative Comparison.** We compare our method against the baselines in terms of foreground similarity (higher is better), object traces (lower is better) and realism (lower is better). As can be seen, DiffEditor [Mou et al. 2024] and DragonDiffusion [Mou et al. 2023] struggle with object traces as they suffer from the object traces issue. PBE [Yang et al. 2022], Anydoor [Chen et al. 2023a], DragDiffusion [Shi et al. 2023] and Diffusion SG [Epstein et al. 2023] struggle with foreground similarity as they tend not to drag the object. In contrast, our method significantly outperforms all the baselines in terms of object traces and also achieves higher foreground similarity with comparable image realism.

Method	Foreground (↑)	Traces (↓)	Realism (↓)
PBE [Yang et al. 2022]	0.614	0.446	0.0029
DiffusionSG [Epstein et al. 2023]	0.515	0.459	0.0096
Anydoor [Chen et al. 2023a]	0.684	0.454	0.0035
DragDiffusion [Shi et al. 2023]	0.738	0.603	0.0011
DragonDiffusion [Mou et al. 2023]	0.818	0.773	0.0009
DiffEditor [Mou et al. 2024]	0.826	0.604	**0.0008**
DiffUHaul (ours)	**0.835**	**0.32**	**0.0008**

Table 2. **Ablation Study.** We ablate the following components of our method: (1) w/o gated self-attention (GSA) masking, (2) w/o self-attention (SA) sharing, (3) w/o soft attention anchoring and (4) w/o DDPM noising. As can be seen, removing the (1) GSA masking harms the foreground similarity, as leakages from neighboring blobs can interfere. Removing the (2) SA sharing or the (3) soft attention anchoring harms the foreground similarity as well, as it reduces the similarity the input image. Removing the DDPM SA bucketing slightly improves the object traces but significantly harms the foreground similarity, as the details of source images are not well preserved.

Method	Foreground (↑)	Traces (↓)	Realism (↓)
DiffUHaul (ours)	**0.835**	0.320	**0.0008**
w/o GSA masking	0.823	0.320	**0.0008**
w/o SA sharing	0.755	0.314	0.0014
w/o soft attention anchoring	0.780	0.322	0.0008
w/o DDPM SA bucketing	0.675	**0.298**	0.0029

along with the above extracted blobs, are then passed to the localized model to get self-attention outputs in every attention layer, as needed. Note that the DDPM self-attention bucketing is specifically designed for the object dragging task, where we aim to preserve the visual details of the real image. It may not be suitable for other image editing tasks that change the object appearance or category.

For extracting the blobs representations from real images, we utilize ODISE [Xu et al. 2023] to get instance segmentation maps, then we use an ellipse fitting optimization with the goal of maximizing the Intersection Over Union (IOU) between the ellipse and the generated mask. Finally, we crop a local region around each blob and use LLaVA-1.5 [Liu et al. 2023] for the local captioning.

Finally, in order to better preserve the background, we incorporated the Blended Latent Diffusion [Avrahami et al. 2023b, 2022] method into our process in which the background pixels are being integrated into the diffusion process in order to seamlessly blend the generated result in the original scene. For more details, please refer to the supplementary material.

5 EXPERIMENTS

In Section 5.1, we compare our method against several baselines, both qualitatively and quantitatively. Next, in Section 5.2 we describe the user study on various methods and present the outcome. Lastly, in Section 5.3, we show the ablation study results to highlight the importance of each component.

5.1 Qualitative and Quantitative Comparison

We compared our method against the most relevant available object dragging baselines. Paint-By-Example (PBE) [Yang et al. 2022] and AnyDoor [Chen et al. 2023a] present a way of adding an object to an image. To use them for object dragging, we crop the object from the source image, apply the image inpaintng in the cropped region, and then add the object in the new location. Diffusion Self-Guidance (Diffusion SG) [Epstein et al. 2023] tackles the general image editing tasks via attention guidance, which can be tailored to object dragging. DragDiffusion [Shi et al. 2023] is designed for the task of keypoint-based dragging, which we can convert into object dragging by selecting multiple points on the source object. Finally, DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] directly tackle the problem of object dragging. For more details, please see the supplementary material.

As can be seen in Figure 8, PBE [Yang et al. 2022], Anydoor [Chen et al. 2023a] and DiffusionSG [Epstein et al. 2023] can hardly preserve the appearance of the edited object and always have undesirable objects or artifacts left in the source location, indicating that existing general-purpose image editing methods tend to completely fail in the object dragging task. DragDiffusion [Shi et al. 2023] struggles with moving the object to the target location, while DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] often suffers from the object traces issue, where the object appears in both source and target locations. In contrast, our method strikes the best balance between effectively dragging the object to the right position and preserving its visual appearance.

To quantify the performance of our method and baselines, we prepare a specialized evaluation dataset based on the COCO [Lin et al. 2014] validation set. We first filter it to contain only images that have a single "thing" object with a prominent size. Then, we use the same blobs extraction pipeline as explained in Section 4.3. For the object dragging task, we randomly sample a new location in the pixel space as the center of the target blob B_d. For each sample, we calculate 8 different target drag locations, resulting in a total dataset of 6,048 samples. For more details, please read the supplementary material. In Figure 9, we provide a qualitative comparison on the automatic dataset, where we make similar observations as before.

Based on this new dataset, we propose three evaluation metrics: *foreground similarity*, *object traces* and *realism*. Foreground similarity quantifies whether the source object indeed dragged to the target location without appearance changes. To this end, we crop a tight box area around the source blob B_s in the source image I_s and around the target blob B_d in the target image I', respectively, and pass the crops to DINOv2 [Oquab et al. 2023] to measure the perceptual similarity after aligning them to a canonical position and masking the background. We strive to *maximize* this metric. To measure the object traces phenomenon, we crop a tight box area around the

Table 3. **User Study.** We compare our method against the baselines using the standard two-alternative forced-choice format. Users were asked to rate which editing result is better (Ours vs. the baseline) in terms of: (1) dragging the object to the desired location (2) leaving no traces of the original object, (3) realism and (4) overall edit quality. The number represents the win rate of our method over each of the baselines. As we can see, our method wins the baselines in all terms more than the random win rate of 50%.

Ours vs	Dragging (↑)	No traces (↑)	Realism (↑)	Overall (↑)
PBE [Yang et al. 2022]	82.14%	78.57%	82.58%	79.68%
DiffusionSG [Epstein et al. 2023]	79.46%	76.78%	77.45%	77.45%
Anydoor [Chen et al. 2023a]	81.02%	76.78%	81.91%	81.25%
DragDiffusion [Shi et al. 2023]	64.73%	58.92%	59.82%	61.16%
DragonDiffusion [Mou et al. 2023]	77.00%	75.22%	81.02%	76.56%
DiffEditor [Mou et al. 2024]	73.43%	70.08%	70.08%	76.33%

source blob B_s in the source image I_s and around the source blob B_s in the target image I'. Next, we masked the target blob B_d area in the target image I'. Similarly, we utilize DINOv2 [Oquab et al. 2023] to measure the perceptual similarity between the crops. We strive to *minimize* this metric. Lastly, to measure the realism of the edited image, we utilized KID score [Binkowski et al. 2018] of sets of 672 real and generated images. For more details, please read the supplementary material.

As can be seen in Table 1, DiffEditor [Mou et al. 2024] and DragonDiffusion [Mou et al. 2023] rank high in object traces as they suffer from object traces problem. PBE [Yang et al. 2022], Anydoor [Chen et al. 2023a], DragDiffusion [Shi et al. 2023] and Diffusion SG [Epstein et al. 2023] struggle with foreground similarity as they tend not to drag the object. On the other hand, our method significantly outperforms all the baselines in terms of object traces, which demonstrates the robustness of our method. In addition, it achieves higher foreground similarity and is on par in terms of image realism. These results are supported by the qualitative comparison.

5.2 User Study

We conduct an extensive user study using the Amazon Mechanical Turk (AMT) platform [Amazon 2024], where the test examples are also sampled from the automatically extracted dataset as explained in Section 5.1. We compare all the baselines using the standard two-alternative forced-choice format. Users were given the source image, the edit instructions and two edited images: one from our method and another one from a baseline. For each comparison, users were asked to rate which edited image is better in terms of: (1) dragging the object to the desired location (2) leaving no traces of the original object, (3) realism and (4) overall edit quality (i.e., taking all the aspects into account). As can be seen in Table 3, our method is preferred over all the baselines in terms of the overall edit quality and different individual perspectives. This observation aligns well with our automatic metrics. The user study suggests that DragDiffusion is the second-strongest baseline, it may be due to the fact that it also result with realistic images, as it also avoids leaving traces of the dragged object, which the automatically calculated KID do not take into account. For more details and statistical significance analysis, please read the supplementary material.

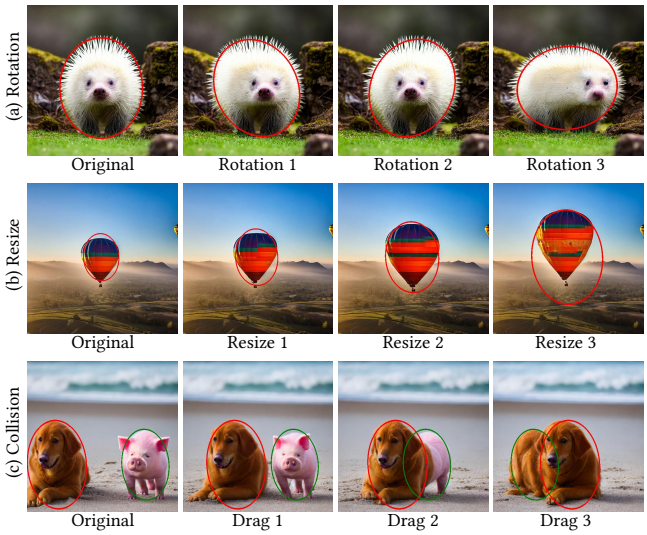

Fig. 7. **Limitations.** Our method suffers from the following limitations: (a) We found our method to be incapable of rotating objects, and instead stretch the object to fit the new blob shape without changing the orientation. (b) We found our method to struggle with resizing object, especially in large resizes (e.g., Resize 3 in the second row). (c) We found our method to struggle to handle colliding object while dragging, which may result with a hybrid between the objects (e.g., Drag 2 in the third row) or one object being merged (e.g., Drag 3 in the third row).

5.3 Ablation Study

We perform the ablation study for the following components of our method: (1) *Without gated self-attention masking* — we remove the gated self-attention masking that is described in Section 4.1. (2) *Without self-attention sharing* — we remove the self-attention sharing component. (3) *Without soft attention anchoring* — we remove the soft attention anchoring that is described in Section 4.2. (4) *Without DDPM noising* — we replace the DDPM noising that is described in Section 4.3 with a DDIM inversion [Song et al. 2020].

We use the same automatic evaluation metrics as described in Section 5.1 to quantify the importance of each component. As can be seen in Table 2, removing the (1) GSA masking harms the foreground similarity, as leakages from neighboring blobs can interfere the visual appearances of the focused object. Removing the (2) SA sharing or the (3) soft attention anchoring harms the foreground similarity as well, as it reduces the similarity to the input image. Removing the DDPM noising slightly improves the object traces, but it significantly harms the foreground similarity, as the reconstructed image itself has changed significantly. For a qualitative visualization of the ablation study, please refer to the supplementary material.

6 LIMITATIONS AND CONCLUSIONS

Our method suffers from the following limitations that are depicted in Figure 7: (a) We found our diffusion anchoring technique introduced in Section 4.2 to be incapable of rotating objects, and instead, as can be seen in Figure 7(a), stretch the object to fit the new blob shape without changing the orientation, this may be caused due

to the fact that rotation involves understanding the 3D structure, which is not reflected by the self-attention nearest-neighbor copying. (b) We found our method to struggle with resizing object, especially in large resizes, as can be seen in Figure 7(b) Resize 3. (c) We found our method to struggle to handle colliding object while dragging, which may result with a hybrid between the objects (Figure 7(c) Drag 2) or one object being merged (Figure 7(c) Drag 3).

In conclusion, we presented DiffUHall, our solution to the seemingly straightforward task of object dragging. We demonstrated that the spatial understanding of the *localized* BlobGEN can be harnessed to this task, using our novel diffusion anchoring technique that manages to merge the location signal from the model with the object appearance signal from the input image. We hope that our contributions will serve as a valuable tools not only for object dragging but also for other creative tasks in the future.

Fig. 8. **Qualitative Comparison.** We compared our method against several baselines on both generated (first three columns) and real images (second three columns). The source and target locations are denoted by red and green points, respectively. As can be seen, PBE [Yang et al. 2022], DiffusionSG [Epstein et al. 2023] and Anydoor [Chen et al. 2023a] mainly suffer from a bad preservation of the foreground object. DragDiffusion [Shi et al. 2023] struggles with dragging the object, while DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] suffers from object traces. Our method, on the other hand, strikes the balance between dragging the object and preserving its identity.

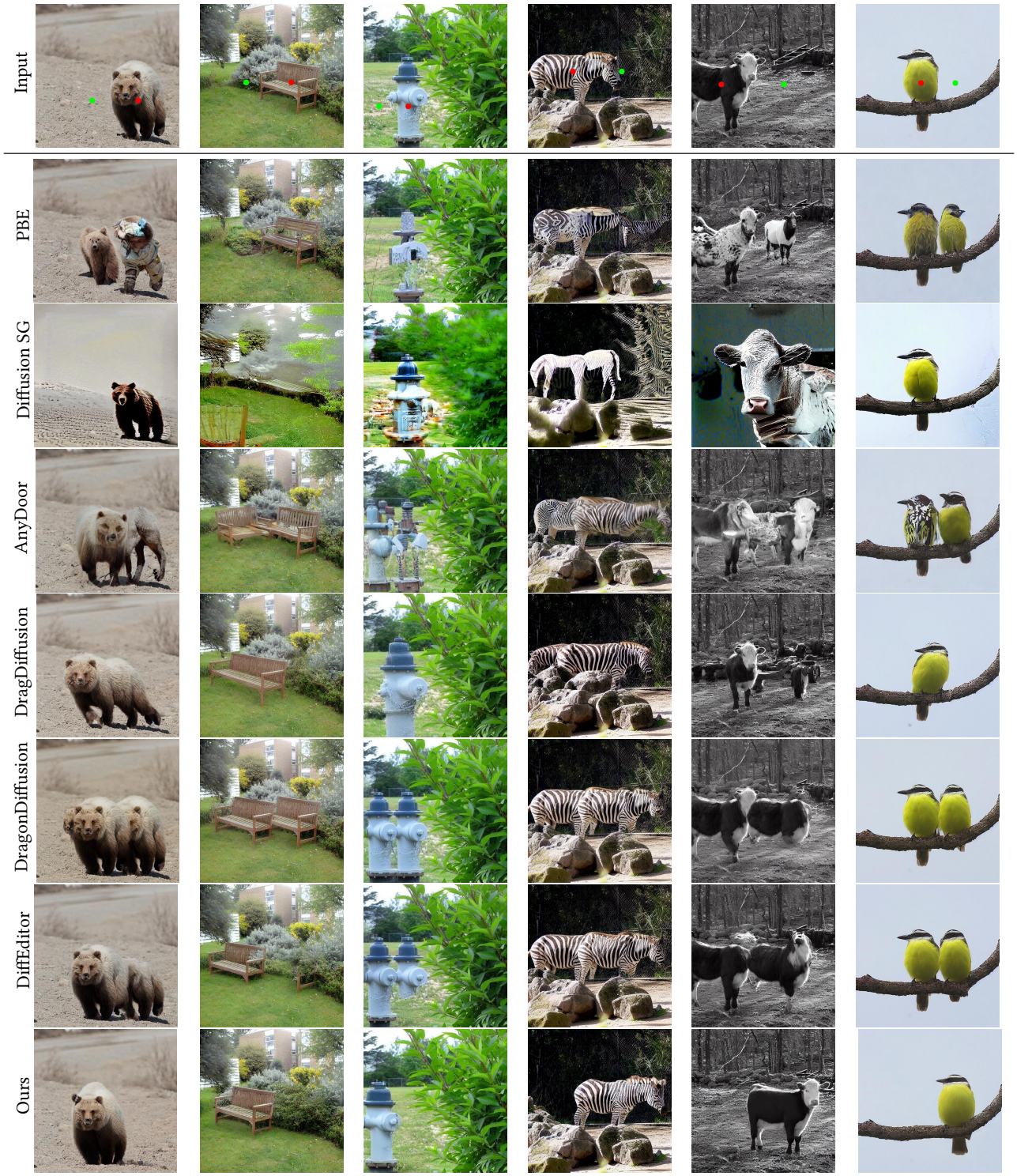

Fig. 9. **Qualitative Automatic Comparison.** As explained in Section 5.1, we used a filtered version of COCO validation set [Lin et al. 2014]. The source and target locations are denoted by red and green points, respectively. As can be seen, PBE [Yang et al. 2022], DiffusionSG [Epstein et al. 2023] and Anydoor [Chen et al. 2023a] mainly suffer from a bad preservation of the foreground object. DragDiffusion [Shi et al. 2023] struggles with dragging the object, while DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] suffers from object traces. Our method, on the other hand, strikes the balance between dragging the object and preserving its identity.

REFERENCES

Yuval Alaluf, Elad Richardson, Gal Metzer, and Daniel Cohen-Or. 2023. A Neural Space-Time Representation for Text-to-Image Personalization. *ArXiv* abs/2305.15391 (2023). https://api.semanticscholar.org/CorpusID:258866047

Amazon. 2024. Amazon Mechanical Turk. https://www.mturk.com/.

Moab Arar, Rinon Gal, Yuval Atzmon, Gal Chechik, Daniel Cohen-Or, Ariel Shamir, and Amit H Bermano. 2023. Domain-agnostic tuning-encoder for fast personalization of text-to-image models. *arXiv preprint arXiv:2307.06925* (2023).

Moab Arar, Andrey Voynov, Amir Hertz, Omri Avrahami, Shlomi Fruchter, Yael Pritch, Daniel Cohen-Or, and Ariel Shamir. 2024. PALP: Prompt Aligned Personalization of Text-to-Image Models. (2024).

Omri Avrahami, Kfir Aberman, Ohad Fried, Daniel Cohen-Or, and Dani Lischinski. 2023a. Break-A-Scene: Extracting Multiple Concepts from a Single Image. *ArXiv* abs/2305.16311 (2023). https://api.semanticscholar.org/CorpusID:258888228

Omri Avrahami, Ohad Fried, and Dani Lischinski. 2023b. Blended Latent Diffusion. *ACM Trans. Graph.* 42, 4, Article 149 (jul 2023), 11 pages. https://doi.org/10.1145/3592450

Omri Avrahami, Thomas Hayes, Oran Gafni, Sonal Gupta, Yaniv Taigman, Devi Parikh, Dani Lischinski, Ohad Fried, and Xi Yin. 2023c. SpaText: Spatio-Textual Representation for Controllable Image Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18370–18380.

Omri Avrahami, Amir Hertz, Yael Vinker, Moab Arar, Shlomi Fruchter, Ohad Fried, Daniel Cohen-Or, and Dani Lischinski. 2023d. The Chosen One: Consistent Characters in Text-to-Image Diffusion Models. *ArXiv* abs/2311.10093 (2023). https://api.semanticscholar.org/CorpusID:265221238

Omri Avrahami, Dani Lischinski, and Ohad Fried. 2022. Blended Diffusion for Text-Driven Editing of Natural Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18208–18218.

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. 2023. Multidiffusion: Fusing diffusion paths for controlled image generation. (2023).

Mikolaj Binkowski, Danica J. Sutherland, Michal Arbel, and Arthur Gretton. 2018. Demystifying MMD GANs. *ArXiv* abs/1801.01401 (2018). https://api.semanticscholar.org/CorpusID:3531856

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. 2023. InstructPix2Pix: Learning to Follow Image Editing Instructions. In *CVPR*.

Mingdeng Cao, Xintao Wang, Zhongang Qi, Ying Shan, Xiaohu Qie, and Yinqiang Zheng. 2023. MasaCtrl: Tuning-Free Mutual Self-Attention Control for Consistent Image Synthesis and Editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 22560–22570.

Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. 2023. Attend-and-Excite: Attention-Based Semantic Guidance for Text-to-Image Diffusion Models. *ACM Transactions on Graphics (TOG)* 42 (2023), 1 – 10. https://api.semanticscholar.org/CorpusID:256416326

Minghao Chen, Iro Laina, and Andrea Vedaldi. 2023b. Training-Free Layout Control with Cross-Attention Guidance. *arXiv preprint arXiv:2304.03373* (2023).

Xi Chen, Lianghua Huang, Yu Liu, Yujun Shen, Deli Zhao, and Hengshuang Zhao. 2023a. AnyDoor: Zero-shot Object-level Image Customization. *ArXiv* abs/2307.09481 (2023). https://api.semanticscholar.org/CorpusID:259951373

Prafulla Dhariwal and Alex Nichol. 2021. Diffusion Models Beat GANs on Image Synthesis. *ArXiv* abs/2105.05233 (2021). https://api.semanticscholar.org/CorpusID:234357997

Yuki Endo. 2022. User-Controllable Latent Transformer for StyleGAN Image Layout Editing. *Computer Graphics Forum* 41 (2022). https://api.semanticscholar.org/CorpusID:251881740

Dave Epstein, Allan Jabri, Ben Poole, Alexei Efros, and Aleksander Holynski. 2023. Diffusion self-guidance for controllable image generation. *Advances in Neural Information Processing Systems* 36 (2023), 16222–16239.

Dave Epstein, Taesung Park, Richard Zhang, Eli Shechtman, and Alexei A. Efros. 2022. BlobGAN: Spatially Disentangled Scene Representations. *ArXiv* abs/2205.02837 (2022). https://api.semanticscholar.org/CorpusID:248524853

Weixi Feng, Xuehai He, Tsu-Jui Fu, Varun Jampani, Arjun Reddy Akula, Pradyumna Narayana, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2022. Training-Free Structured Diffusion Guidance for Compositional Text-to-Image Synthesis. In *The Eleventh International Conference on Learning Representations*.

Yarden Frenkel, Yael Vinker, Ariel Shamir, and Daniel Cohen-Or. 2024. Implicit Style-Content Separation using B-LoRA. *ArXiv* abs/2403.14572 (2024). https://api.semanticscholar.org/CorpusID:268553753

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit Haim Bermano, Gal Chechik, and Daniel Cohen-or. 2022. An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. In *The Eleventh International Conference on Learning Representations*.

Rinon Gal, Moab Arar, Yuval Atzmon, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. 2023. Encoder-based domain tuning for fast personalization of text-to-image models. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–13.

Michal Geyer, Omer Bar-Tal, Shai Bagon, and Tali Dekel. 2023. Tokenflow: Consistent diffusion features for consistent video editing. *arXiv preprint arXiv:2307.10373* (2023).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

Amir Hertz, Kfir Aberman, and Daniel Cohen-Or. 2023. Delta denoising score. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2328–2337.

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

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

Jonathan Ho. 2022. Classifier-Free Diffusion Guidance. *ArXiv* abs/2207.12598 (2022). https://api.semanticscholar.org/CorpusID:249145348

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Proc. NeurIPS*.

Eliahu Horwitz, Jonathan Kahana, and Yedid Hoshen. 2024. Recovering the Pre-Fine-Tuning Weights of Generative Models. *ArXiv* abs/2402.10208 (2024). https://api.semanticscholar.org/CorpusID:267682124

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2021. Alias-Free Generative Adversarial Networks. arXiv:2106.12423 [cs.CV]

Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4401–4410.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8110–8119.

Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. 2023. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6007–6017.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. 2023. GLIGEN: Open-Set Grounded Text-to-Image Generation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), 22511–22521. https://api.semanticscholar.org/CorpusID:255942528

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved Baselines with Visual Instruction Tuning. *ArXiv* abs/2310.03744 (2023). https://api.semanticscholar.org/CorpusID:263672058

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. 2021. SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations. In *International Conference on Learning Representations*.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2023. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6038–6047.

Eyal Molad, Eliahu Horwitz, Dani Valevski, Alex Rav Acha, Y. Matias, Yael Pritch, Yaniv Leviathan, and Yedid Hoshen. 2023. Dreamix: Video Diffusion Models are General Video Editors. *ArXiv* abs/2302.01329 (2023).

Chong Mou, Xintao Wang, Jie Song, Ying Shan, and Jian Zhang. 2023. DragonDiffusion: Enabling Drag-style Manipulation on Diffusion Models. *ArXiv* abs/2307.02421 (2023). https://api.semanticscholar.org/CorpusID:259342813

Chong Mou, Xintao Wang, Jie Song, Ying Shan, and Jian Zhang. 2024. DiffEditor: Boosting Accuracy and Flexibility on Diffusion-based Image Editing. *ArXiv* abs/2402.02583 (2024). https://api.semanticscholar.org/CorpusID:267499649

Weili Nie, Sifei Liu, Morteza Mardani, Chao Liu, Benjamin Eckart, and Arash Vahdat. 2024. Compositional Text-to-Image Generation with Dense Blob Representations. arXiv:2405.08246 [cs.CV]

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Q. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russ Howes, Po-Yao (Bernie) Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Huijiao Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. 2023. DINOv2: Learning Robust Visual Features without Supervision. *ArXiv* abs/2304.07193 (2023). https://api.semanticscholar.org/CorpusID:258170077

Xingang Pan, Ayush Kumar Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. 2023. Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold. *ACM SIGGRAPH 2023 Conference Proceedings* (2023). https://api.semanticscholar.org/CorpusID:258762550

Or Patashnik, Daniel Garibi, Idan Azuri, Hadar Averbuch-Elor, and Daniel Cohen-Or. 2023. Localizing Object-level Shape Variations with Text-to-Image Diffusion

Models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), 22994–23004. https://api.semanticscholar.org/CorpusID:257632209

Quynh Phung, Songwei Ge, and Jia-Bin Huang. 2023. Grounded Text-to-Image Synthesis with Attention Refocusing. *arXiv preprint arXiv:2306.05427* (2023).

Dustin Podell, Zion English, Kyle Lacey, A. Blattmann, Tim Dockhorn, Jonas Muller, Joe Penna, and Robin Rombach. 2023. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. *ArXiv* abs/2307.01952 (2023). https://api.semanticscholar.org/CorpusID:259341735

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. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning*.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125* (2022).

Elad Richardson, Kfir Goldberg, Yuval Alaluf, and Daniel Cohen-Or. 2023. ConceptLab: Creative Generation using Diffusion Prior Constraints. *arXiv preprint arXiv:2308.02669* (2023).

Robin Rombach, A. Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. High-Resolution Image Synthesis with Latent Diffusion Models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 10674–10685.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 22500–22510.

Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. 2023. Emu Edit: Precise Image Editing via Recognition and Generation Tasks. *ArXiv* abs/2311.10089 (2023). https://api.semanticscholar.org/CorpusID:265221391

Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent Y. F. Tan, and Song Bai. 2023. DragDiffusion: Harnessing Diffusion Models for Interactive Point-based Image Editing. *ArXiv* abs/2306.14435 (2023). https://api.semanticscholar.org/CorpusID:259252555

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*. PMLR, 2256–2265.

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*.

Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems* 32 (2019).

Yoad Tewel, Omri Kaduri, Rinon Gal, Yoni Kasten, Lior Wolf, Gal Chechik, and Yuval Atzmon. 2024. Training-Free Consistent Text-to-Image Generation. *ArXiv* abs/2402.03286 (2024). https://api.semanticscholar.org/CorpusID:267412997

Narek Tumanyan, Michal Geyer, Shai Bagon, and Tali Dekel. 2023. Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1921–1930.

Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. 2022. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers.

Andrey Voynov, Q. Chu, Daniel Cohen-Or, and Kfir Aberman. 2023. P+: Extended Textual Conditioning in Text-to-Image Generation. *ArXiv* abs/2303.09522 (2023).

Jianyuan Wang, Ceyuan Yang, Yinghao Xu, Yujun Shen, Hongdong Li, and Bolei Zhou. 2021. Improving GAN Equilibrium by Raising Spatial Awareness. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 11275–11283. https://api.semanticscholar.org/CorpusID:244772988

Sheng-Yu Wang, David Bau, and Jun-Yan Zhu. 2022. Rewriting geometric rules of a GAN. *ACM Transactions on Graphics (TOG)* 41 (2022), 1 – 16. https://api.semanticscholar.org/CorpusID:250956766

Daniel Winter, Matan Cohen, Shlomi Fruchter, Yael Pritch, Alex Rav-Acha, and Yedid Hoshen. 2024. ObjectDrop: Bootstrapping Counterfactuals for Photorealistic Object Removal and Insertion. *ArXiv* abs/2403.18818 (2024). https://api.semanticscholar.org/CorpusID:268724005

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Weixian Lei, Yuchao Gu, Wynne Hsu, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. 2022. Tune-A-Video: One-Shot Tuning of Image Diffusion Models for Text-to-Video Generation. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2022), 7589–7599. https://api.semanticscholar.org/CorpusID:254974187

Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. 2023. Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), 2955–2966. https://api.semanticscholar.org/CorpusID:257405338

Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. 2022. Paint by Example: Exemplar-based Image Editing with Diffusion Models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 18381–18391. https://api.semanticscholar.org/CorpusID:253802085

Zhengyuan Yang, Jianfeng Wang, Zhe Gan, Linjie Li, Kevin Lin, Chenfei Wu, Nan Duan, Zicheng Liu, Ce Liu, Michael Zeng, et al. 2023. Reco: Region-controlled text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14246–14255.

Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. 2015. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *ArXiv* abs/1506.03365 (2015). https://api.semanticscholar.org/CorpusID:8317437

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. 2022. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *arXiv preprint arXiv:2206.10789* (2022).

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 3836–3847.

Guangcong Zheng, Xianpan Zhou, Xuewei Li, Zhongang Qi, Ying Shan, and Xi Li. 2023. LayoutDiffusion: Controllable Diffusion Model for Layout-to-image Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 22490–22499.

A IMPLEMENTATION DETAILS

In Appendix A.1 we start by providing the implementation details of our method. Next, in Appendix A.2 we provide the baselines' implementation details. Later, in Appendix A.3 we provide the implementation details of the automatic metrics we used. Finally, in Appendix A.4 we provide the detail of the user study we conducted.

A.1 Implementation Details of Our Method

Below, we provide the full implementation details of our method: in Appendix A.1.1 we start by providing the implementation details for the gated self-attention visualization we used, next, in Appendix A.1.2 we provide the implementation details of the soft self-attention anchoring we used, and finally, in Appendix A.1.3 we explain about the Blended Latent Diffusion integration.

A.1.1 Gated Self-Attention Visualization Implementation Details. As explained in Section 4.1, we offer a method to visualize the gated self-attention layer of GLIGEN [Li et al. 2023] and BlobGEN [Nie et al. 2024]. In gated self-attention, a projection layer first converts the CLIP [Radford et al. 2021] text embeddings of the text description S_i to the text tokens $T = \{t_1, ...t_n\}$. They are then merged with the visual tokens $V = \{v_1, ...v_k\}$ into a unified set $V \cup T = \{v_1, ...v_k, t_1, ...t_n\}$, which altogether are used to calculate the self-attention features, using the standard self-attention mechanism (plus a gated skip connection).

We examine the attention between the projected text token t_i and all the visual tokens V. This is a K-dimensional vector, which we first reshape into two dimensions $\sqrt{K} \times \sqrt{K}$, and then resize to a canonical size. We term these maps "reshaped self-attention", which are averaged over all the denoising steps. We average these maps across all the diffusion steps and all the layers, by resizing them to a canonical size.

In Figure 10 we provide a full visualization of these maps. As can be seen, even though there is no constraint on the attention distribution (as in any other self-attention layer), we found empirically that the vast majority of the attention is being formed between the used pooled textual tokens (the first part of T) and the visual tokens V. There is little interaction within the individual sets themselves. This behavior suggests that this kind of self-attention layers behaves as de facto cross attention layers. Our inference-time masking mechanism (right) further constrains each textual token to only attend to its corresponding visual token within the blob region.

Please note that these kind of visualizations and the masking manipulations are different from the those common in the text-to-image diffusion-based models literature [Avrahami et al. 2023a; Chefer et al. 2023; Hertz et al. 2022] as we do not manipulate the traditional *cross-attention* layers but the gated *self-attention* layers.

A.1.2 Soft Self-Attention Anchoring Implementation Details. As explained in Section 4.2, we propose the soft self-attention anchoring to fuse the spatial information from the localized model and the appearance information from the input source image. Specifically, in the first $\rho = \frac{T}{2}$ steps of the denoising process, we perform an adaptive, *soft blending* of the attention features of the generated target image with the features of the source image. The interpolation coefficient is time-dependent: we take more visual appearance from the source image in the beginning but more spatial information from the target image in the later steps.

Next, during the last $T - \rho$ steps of the denoising process, we use the soft blending result O_a as anchor points for the target object. In each denoising step $t \in [T - \rho, ..., 2, 1]$ and each self-attention layer, we perform the *nearest-neighbor copying*: each entry from the anchor attention features O_a within the target blob B_d is replaced by its nearest-neighbor entry from the source attention features O_s within the source blob B_s. To calculate the nearest-neighbor, we normalize each self-attention entry of O and calculate its cosine similarity with each entry of O_s. Please note that the nearest-neighbor operation is only within the source blobs B_s and destination blob B_d boundaries. To calculate the these blobs, we reshape them to the corresponding self-attention size of each layer.

A.1.3 Blended Latent Diffusion Integration. Blended Latent Diffusion [Avrahami et al. 2023b, 2022] is a method designed for localized image editing using text-to-image diffusion models. the input image is fused into the diffusion process along with an input mask to preserve it background, while encouraging the generated content (in the unmasked area) to be consistent to the background. We also use this method in our pipeline of editing real images, as introduced in Section 4.3. Given the source blob B_s and the destination blob B_d provided by the user, we take the union blob that contains both of them $B_u = B_s \cup B_d$, and morphologically dilate it with a kernel of a size of 50×50. We treat this dilated blob as the editable area, which we provide to the Blended Latent Diffusion method to edit real images during the entire diffusion process (i.e. the hyperparameter of noising diffusion steps $k = T$, where T is the total number of diffusion steps).

Table 4. **Inference Time Comparison.** We report the inference time of the baselines and our method of editing a single 512×512 image. All the reported running times we calculated using a single NVIDIA A100 GPU.

Method	Inference time (*sec*)
PBE [Yang et al. 2022]	9 sec
Diffusion SG [Epstein et al. 2023]	14 sec
Anydoor [Chen et al. 2023a]	9 sec
DragDiffusion [Shi et al. 2023]	148 sec
DragonDiffusion [Mou et al. 2023]	13 sec
DiffEditor [Mou et al. 2024]	13 sec
DiffUHaul (ours)	13 sec

A.2 Implementation Details of Baselines

As described in Section 5.1, we compare our method against the available, most relevant object dragging baselines: Paint-By-Example [Yang et al. 2022], AnyDoor [Chen et al. 2023a], Diffusion Self-Guidance [Epstein et al. 2023], DragDiffusion [Shi et al. 2023], DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024]. Out of these, Diffusion Self-Guidance, DragonDiffusion and DiffEditor directly support the task of object dragging. The rest of these baselines need some adaptations to our problem, as described below.

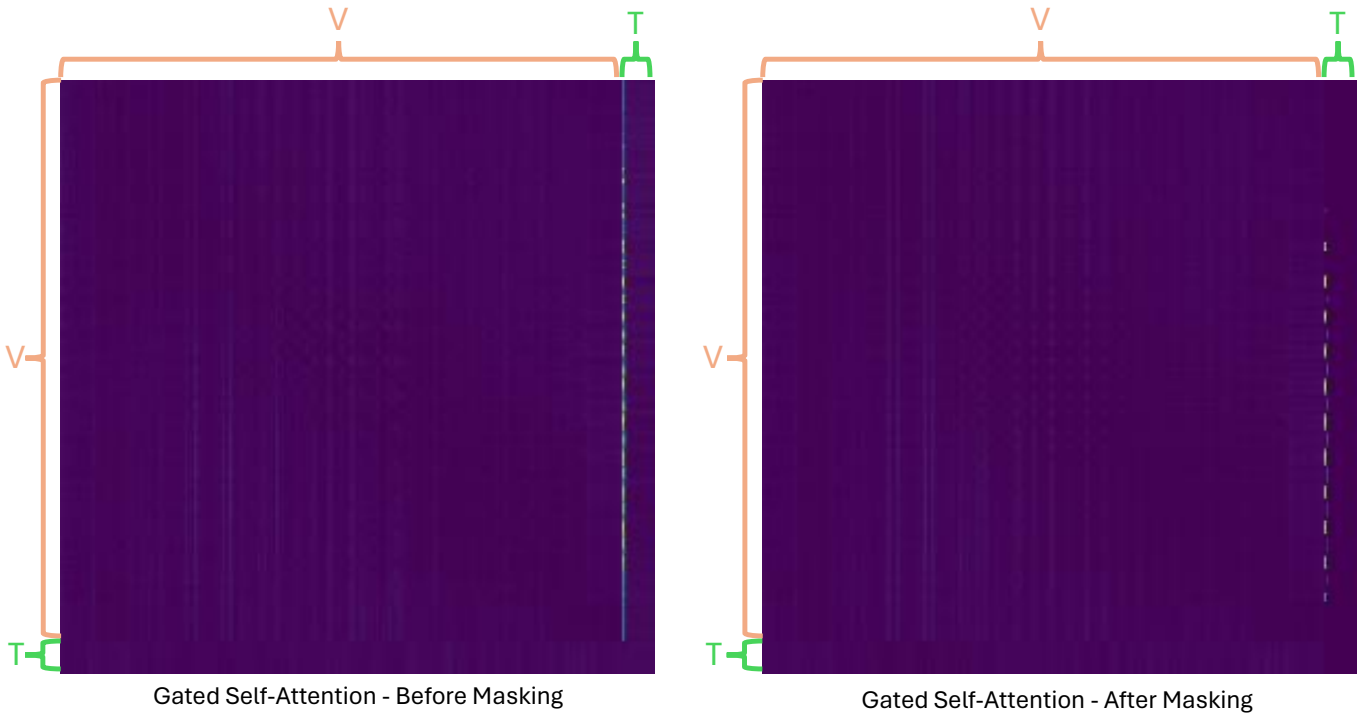

Fig. 10. **Full Gated Self-Attention Visualization.** We provide our visualization for the gated self-attention layers, before our inference time masking (left) and after it (right). As can be seen, even though there is no constraint on the attention distribution (as in any other self-attention layer), we found empirically that the vast majority of the attention is being formed between the used pooled textual tokens (the first part of T) and the visual tokens V. And not between the sets themselves. This behavior suggests that this kind of layers behaves de facto as cross attention layers. Our inference-time masking mechanism (right) constrain the attention between the textual tokens and only their corresponding visual tokens within their blobs.

Paint-By-Example and AnyDoor present a way to add an object to an image. Hence, in order to convert it to our problem setting we constructed a designated pipline: we started taking the source image I and inpaint the source area blob B_s using Stable Diffusion Inpaint [von Platen et al. 2022], to get an inpainted version $\hat{I}$. Then, we used Paint-By-Example/Anydoor to inpaint the new image $\hat{I}$ again, in the target blob area B_d by providing the original object in the original image I as a reference.

DragDiffusion is originally designed to tackle the problem of keypoint-based dragging. Thus, in order to adapt it to our method, we take the centroid of the source blob B_s as well as other points sampled inside the source blob region, and then translate them to the target blob B_d.

We used the official baseline implementations with a comparable backbone of Stable Diffusion v1 [Rombach et al. 2021] using 50 DDIM diffusion steps, except Diffusion Self-Guidance [Epstein et al. 2022], of which the only available implementation is based upon SDXL [Podell et al. 2023].

In Table 4 we report the inference time of our method and the baselines for a single image editing using an NVIDIA A100 GPU. All the methods takes around 10 seconds, except DragDiffusion that is using an extensive LoRA [Horwitz et al. 2024; Hu et al. 2021] training and latent optimization, which increases the inference time significantly.

We used the following third-party packages in this research:

- Official GLIGEN [Li et al. 2023] implementation at https://github.com/gligen/GLIGEN.
- Official Paint-By-Example [Yang et al. 2022] Diffusers [von Platen et al. 2022] implementation.
- Official Diffusion Self-Guidance [Epstein et al. 2023] SDXL implementation at https://colab.research.google.com/drive/1SEM1R9mI9cF-aFpqg3NqHP8gN8irHuJi.
- Official AnyDoor [Chen et al. 2023a] implementation at https://github.com/ali-vilab/AnyDoor.
- Official DragDiffusion [Shi et al. 2023] implementation at https://github.com/Yujun-Shi/DragDiffusion.
- Official DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] at https://github.com/MC-E/DragonDiffusion.
- DINOv2 [Oquab et al. 2023] ViT-g/14 implementation by HuggingFace Transformers [Wolf et al. 2020] at https://github.com/huggingface/transformers.

A.3 Implementation Details of Automatic Metrics

As described in Section 5.1, in order to automatically compare our method against baselines quantitatively, we utilized COCO [Lin et al. 2014] validation dataset. We filtered it to contain only images with a main "thing" class object (the number of "stuff" object is unbounded) that occupies at least 5% of the image size, but not more than 25% of the image size. Then, we utilize ODISE [Xu et al. 2023] to get instance segmentation maps, then we use an ellipse fitting optimization with the goal of maximizing the Intersection Over Union (IOU) between the ellipse and the generated mask. Next, we crop a local region around each blob and use LLaVA-1.5 [Liu et al. 2023] for the local captioning. Finally, we choose a random location for the target blob B_d that is at least 64 pixels. This resulted with 672 filtered images and 8 target blob locations per image, which is a total of 6,048 evaluated samples per baseline.

Next, we propose using three metrics: foreground similarity, object traces and realism. Foreground similarity quantifies whether the source object is indeed dragged to the target location. To this end, we crop a tight square area around the source blob B_s in the source image I, and the target blob B_d in the target image T_t. Next, we align the crops to a canonical position and mask the background in these crops by aligning the object to the left side of the image (in order to avoid translation artifacts). Finally, we utilize DINOv2 [Oquab et al. 2023] to measure the perceptual similarity between these crops. We strive to *maximize* this metric.

Similarly, in order to measure the object duplication phenomenon, we crop a tight square area around the source blob B_s in the source image I, and around the source blob B_s in the target image I'. Next, we mask the target blob B_d area in the target image I'. Finally, we utilize again DINOv2 [Oquab et al. 2023] to measure the perceptual similarity between these crops. We strive to *minimize* this metric. Lastly, in order to measure the realism of the image, we compute the KID score [Binkowski et al. 2018] using 672 real and generated images. The reason of using KID instead of the FID score [Heusel et al. 2017] is that it better aligns with human perception of image generation quality when the provided real and fake sets are small.

A.4 User Study Implementation Details

Table 5. **User Study Statistical Significance.** A binomial statistical test of the user study results suggests that our results are statistically significant (p-value < 5%)

Ours vs	Dragging (↑) p-value	No traces (↑) p-value	Realism (↑) p-value	Overall (↑) p-value
PBE [Yang et al. 2022]	< 1e-8	< 1e-8	< 1e-8	< 1e-8
DiffusionSG [Epstein et al. 2023]	< 1e-8	< 1e-8	< 1e-8	< 1e-8
Anydoor [Chen et al. 2023a]	< 1e-8	< 1e-8	< 1e-8	< 1e-8
DragDiffusion [Shi et al. 2023]	< 1e-8	< 2e-5	< 6e-6	< 5e-7
DragonDiffusion [Mou et al. 2023]	< 1e-8	< 1e-8	< 1e-8	< 1e-8
DiffEditor [Mou et al. 2024]	< 1e-8	< 1e-8	< 1e-8	< 1e-8

As explained in Section 5.2 We conduct an extensive user study using the Amazon Mechanical Turk (AMT) platform [Amazon 2024]. We use the automatically extracted dataset as explained in Section 5.1 in the main paper. We compare all the baselines using the standard two-alternative forced-choice format. Users are instructed the following "In the following image, we are interested in moving the {CATEGORY} from the original location, indicated by the red dot, to the target location, indicated by a green dot." where {CATEGORY} is the [Lin et al. 2014] object class category. Then, the users are given two editing results: our method and one of the baselines, and are asked: (1) "Which of the results is better in moving the {CATEGORY} to the target location?", (2) "Which of the results is better in leaving no traces of the {CATEGORY} in the original location?", (3) "Which of the results looks more realistic?" and (4) "Which of the results is better overall?". The users are also given detailed instructions with examples. An example of one trail can be seen in Figure 11, and the full instructions can be seen in Figure 12.

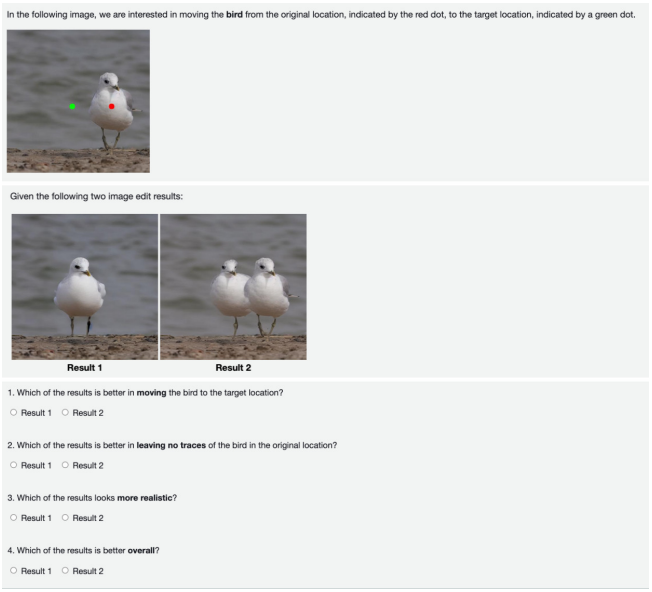

Fig. 11. **User Study Trial.** We provide an example of one trial task in the user study we conducted using Amazon Mechanical Turk (AMT) [Amazon 2024]. The users were asked four questions of a two-alternative forced-choice format. The full instructions can be seen in Figure 12.

We gather 7 ratings per sample, resulting 448 ratings per baseline, totaling 2,688 responses. The time allotted per task is one hour, to allow the raters to properly evaluate the results without time pressure. A binomial statistical test of the user study results, as presented in Table 5, suggesting that our results are statistically significant (p-value < 5%).

B ADDITIONAL RESULTS

In Figure 13 we provide an additional qualitative comparison of our method against the baselines on the automatically extracted dataset (as explained in Appendix A.3). As can be seen, PBE [Yang et al. 2022], DiffusionSG [Epstein et al. 2023] and Anydoor [Chen et al. 2023a] mainly suffer from a bad preservation of the foreground object. DragDiffusion [Shi et al. 2023] struggles with dragging the object, while DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] suffers from object traces. Our method, on the other hand,

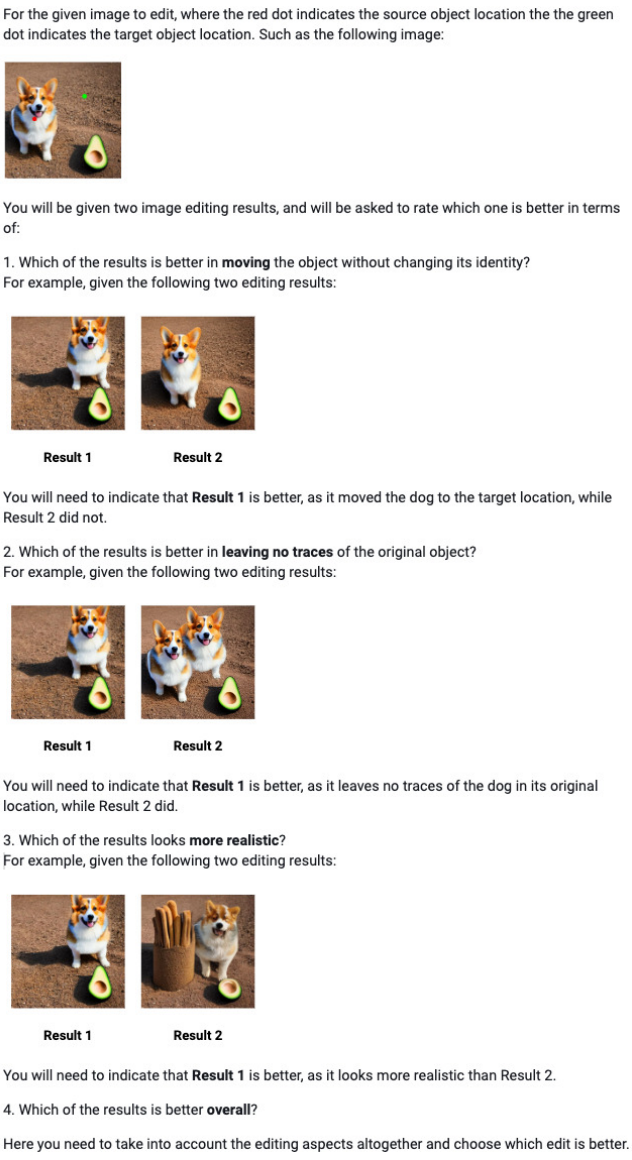

Fig. 12. **User Study Instructions.** We provide the full instructions for the user study we conducted using Amazon Mechanical Turk (AMT) [Amazon 2024], to compare our method with each baseline.

strikes the balance between dragging the object and preserving its identity.

In addition, in Figure 14 we provide a qualitative visualization of the ablation study we conducted. As can be seen, PBE [Yang et al. 2022], DiffusionSG [Epstein et al. 2023] and Anydoor [Chen et al. 2023a] mainly suffer from a bad preservation of the foreground object. DragDiffusion [Shi et al. 2023] struggles with dragging the object, while DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] suffers from object traces. Our method, on the other hand, strikes the balance between dragging the object and preserving its identity.

C SOCIETAL IMPACT

We believe that the advent of technology enabling seamless object dragging within images holds tremendous promise for a wide array of creative and practical uses. It may democratize content manipulation for individuals lacking expertise and artistic skills. Furthermore, we believe that this method may present an invaluable tool for professional artists by expediting their creative processes without compromising quality.

Conversely, akin to other generative AI technologies, this method is susceptible to misuse, potentially giving rise to the creation of deceptive and misleading visual content. The ease and accessibility afforded by this technology could amplify concerns regarding the authenticity and trustworthiness of visual media in various contexts, including journalism, advertising, and social media which may erode the public trust in such content. Therefore, while recognizing its transformative potential, it is imperative to remain vigilant and implement appropriate safeguards to mitigate the proliferation of misinformation and uphold ethical standards in content creation and dissemination.

Fig. 13. **Additional Qualitative Automatic Comparison.** As explained in Section 5.1, we used a filtered version of COCO validation set [Lin et al. 2014]. The source and target locations are denoted by red and green points, respectively. As can be seen, PBE [Yang et al. 2022], DiffusionSG [Epstein et al. 2023] and Anydoor [Chen et al. 2023a] mainly suffer from a bad preservation of the foreground object. DragDiffusion [Shi et al. 2023] struggles with dragging the object, while DragonDiffusion [Mou et al. 2023] and DiffEditor [Mou et al. 2024] suffers from object traces. Our method, on the other hand, strikes the balance between dragging the object and preserving its identity.

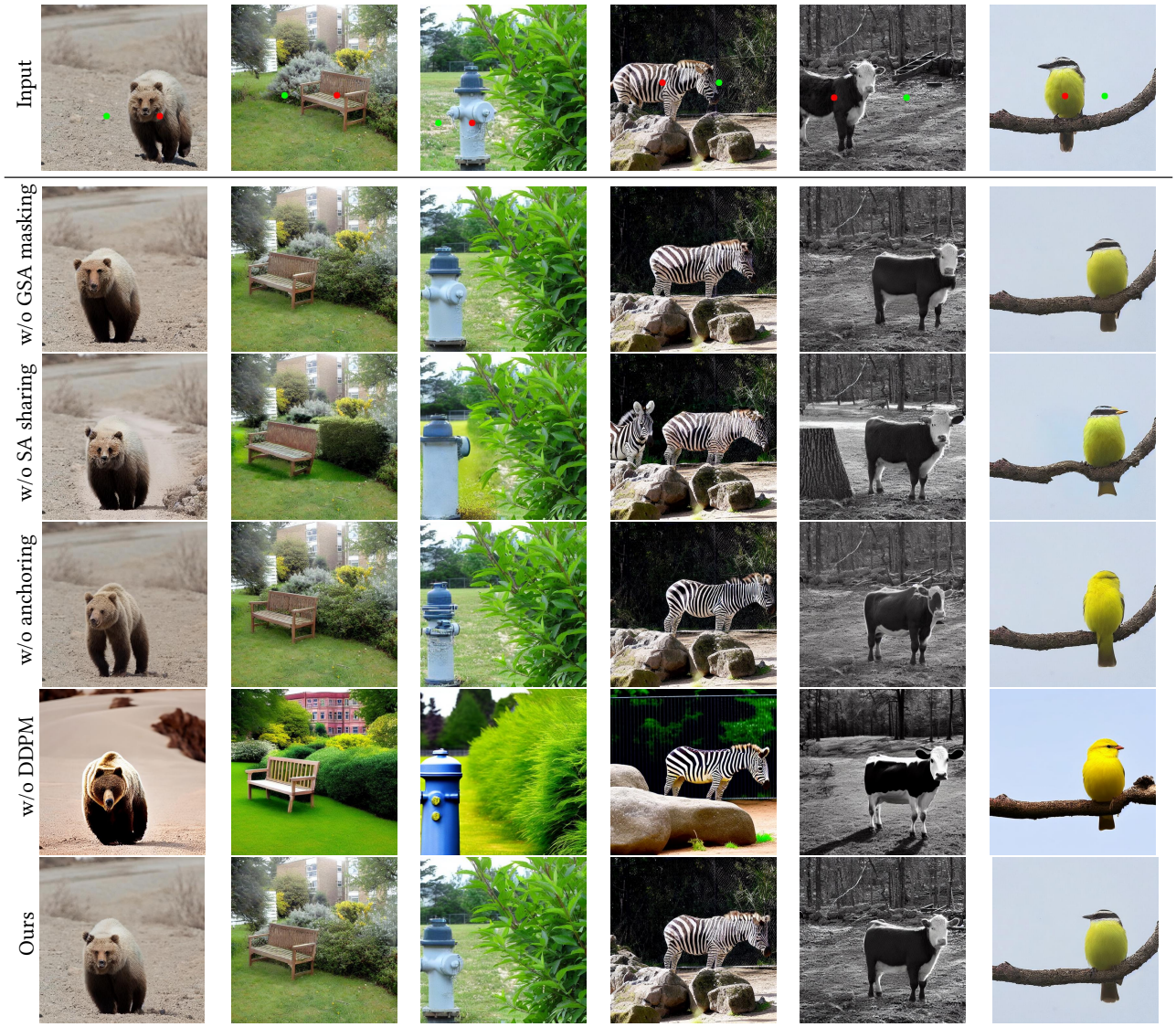

Fig. 14. **Qualitative Results in Ablation Study.** As explained in Section 5.3, we ablate four key components of our method: (a) w/o GSA masking, (b) w/o SA sharing, (c) w/o soft attention anchoring and (d) w/o DDPM SA attention. As can be seen, all these components improve the foreground object consistency.