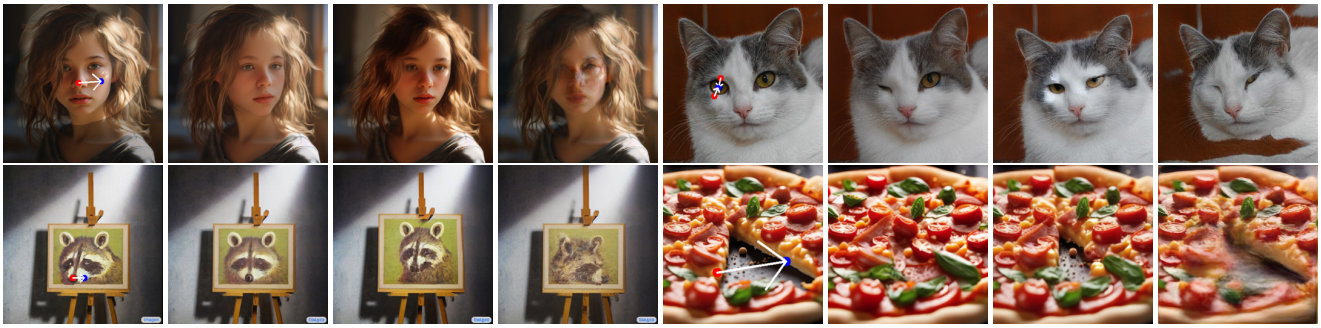
# Drag Your Noise: Interactive Point-based Editing via Diffusion Semantic Propagation

Haofeng Liu[1,2*] Chenshu Xu[1*] Yifei Yang[1] Lihua Zeng[2] Shengfeng He[1†]

[1]Singapore Management University [2]South China Normal University



(a) Input (b) DragNoise (c) DragDiffusion (d) DragGAN (e) Input (f) DragNoise (g) DragDiffusion (h) DragGAN

Figure 1. We present DragNoise, a point-based interactive editing framework that avoids the global adjustments of the latent code/map, a common issue in frameworks like DragGAN [23] and DragDiffusion [29], facilitating stable and semantically accurate point-based editing.

## Abstract

*Point-based interactive editing serves as an essential tool to complement the controllability of existing generative models. A concurrent work, DragDiffusion, updates the diffusion latent map in response to user inputs, causing global latent map alterations. This results in imprecise preservation of the original content and unsuccessful editing due to gradient vanishing. In contrast, we present DragNoise, offering robust and accelerated editing without retracing the latent map. The core rationale of DragNoise lies in utilizing the predicted noise output of each U-Net as a semantic editor. This approach is grounded in two critical observations: firstly, the bottleneck features of U-Net inherently possess semantically rich features ideal for interactive editing; secondly, high-level semantics, established early in the denoising process, show minimal variation in subsequent stages. Leveraging these insights, DragNoise edits diffusion semantics in a single denoising step and efficiently propagates these changes, ensuring stability and efficiency in diffusion editing. Comparative experiments reveal that DragNoise achieves superior control and semantic retention, reducing the optimization time by over 50% compared to DragDiffusion. Our codes are available at https://github.com/haofengl/DragNoise.*

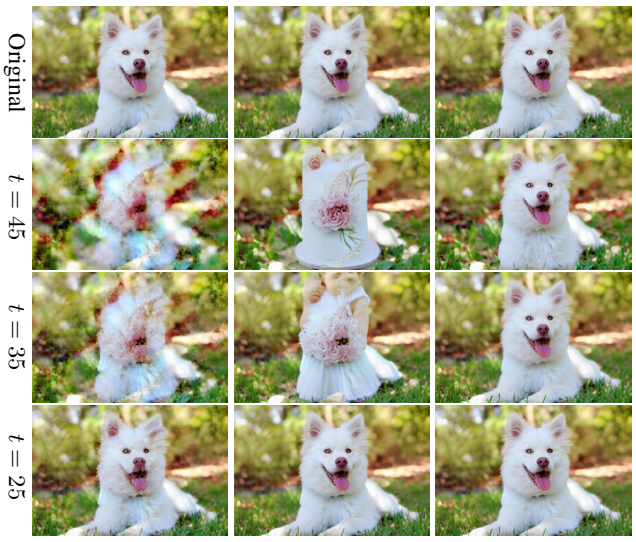
## 1. Introduction

The limited controllability inherent in diffusion models [7, 11] highlights the need for interactive editing in image manipulation. Consequently, recent advancements have led to a variety of interactive approaches. These include text-guided editing [3, 5, 10, 16, 17], stroke-based editing [20], and exemplar-based methods [9, 13, 39, 42]. As the demand for more user-friendly and precise editing methods grows, the implementation of drag-and-drop manipulation of control points has emerged as a straightforward and efficient approach in real-world applications.

In the field of point-based image editing, DragGAN [23] represents a significant milestone by leveraging generative adversarial networks (GANs) [8, 15]. Despite its innovation, the inherent constraints of GANs often limit achieving high-quality edited outcomes. Additionally, GAN-based editing methods [1, 36, 38, 44, 45], which involve optimizing a new latent code corresponding to the edited result, struggle with preserving global content, as illustrated in Figs. 1d and 1h. Different from traditional "outer-inversion" that converts a real image into a latent code, we term this internal optimization process of inverting user editing into the latent code as "inner-inversion" (see Fig. 4a). A concurrent work, DragDiffusion [29] progresses this field by applying diffusion models, capitalizing on the strengths of large-scale pre-trained models. Although DragDiffusion applies diffusion models, it adheres to the concept of "inner-inversion". This method guides the optimization of intermediate noisy latent maps to generate outputs that reflect the intended editing (Fig. 4b).

However, two main issues emerge with DragDiffusion: **gradient vanishing** and **inversion fidelity**. Gradient vanishing during optimization occurs due to the reliance on motion

*The first two authors contributed equally.
†Corresponding author (*shengfenghe@smu.edu.sg*).




(a) Encoder Block 1 (b) Encoder Block 3 (c) Bottleneck

Figure 2. Reconstructed images by DDIM inversion, where features of different levels (column) are copied to corresponding layers in all subsequent U-Nets, beginning from various denoising timesteps (row). The original images, reconstructed without feature copying, are provided for comparison. This synchronization of bottleneck features across all subsequent steps reveals that the core semantics of the diffusion process are encoded within the bottleneck layer, predominantly learned in the early phases of the denoising process.

supervision loss, which is based on feature differences before and after dragging. This issue is exacerbated when feature differences are minimal and the back-propagation chain in inversion is lengthy, leading to "under-dragging" in their results, as evident in the bottom of Figs. 1c and 1g. Additionally, maintaining reconstruction fidelity remains a longstanding challenge in inversion techniques. Although DragDiffusion [29] improves spatial control by extending "inner-inversion" to 2D latent maps, surpassing DragGAN's optimization of 1D latent codes, it still struggles in fine reconstruction due to its optimization path back to the noisy latent map, as demonstrated in the top of Figs. 1c and 1g.

**Diffusion Semantic Analysis.** Prior studies [4, 34] have demonstrated that intermediate diffusion features from the noise predictor not only effectively capture semantic information but also facilitate structure-to-appearance controllability. Inspired by them, we reevaluate the necessity of retracing the latent map and explore diffusion semantics within the editing mechanism. Initially, we conduct feature analysis using DDIM inversion [7, 32] on the pre-trained Stable Diffusion [26] model. To understand the semantics learned by different U-Net layers, we copy features from these layers (shown in different columns) and replace the corresponding ones in all subsequent U-Nets, starting from various denoising timesteps (shown in different rows). This is to show *where* and *when* diffusion models learn semantic knowledge. The resulting images and numerical results are shown in Fig. 2 and Fig. 3 respectively.

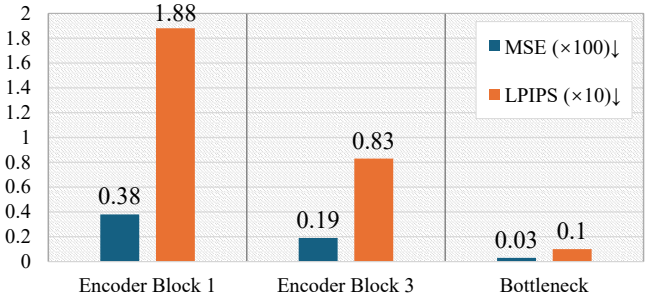



Figure 3. Quantitative analysis on middle-block feature replacement. This involves replacing features at all subsequent timesteps with those from timestep 35, using features from various layers. Our evaluation metrics, MSE and LPIPS [43] were used to compare the reconstructed images against the original inputs.

As the U-Net encoder progressively shrinks the features, higher-level features are attained as the network approaches the bottleneck. The reconstructed images demonstrate that the substitution of low-level features ("Encoder Block 1" and "Encoder Block 3") compromises image reconstruction detail and quality. This outcome is attributed to the distinct roles of low-level features at different timesteps that cannot be shared, such as adding fine textures in the final stages. On the other hand, the bottleneck, producing the high-level feature in the noise predictor, exhibits the capacity to capture more complete semantics even during early timesteps. Generally, copying and replacing features at later timesteps yield better reconstruction of the original image. However, at an early timestep 45, the bottleneck feature can encapsulate the rough outline of the dog, albeit missing finer details like ears and legs. Interestingly, substituting the bottleneck feature from timestep 35 preserves the overall structure, and propagating this early timestep semantic to subsequent steps does not diminish reconstruction quality. These findings lead us to conclude that the bottleneck feature represents an optimal diffusion semantic representation, particularly suitable for efficient editing. Since it can be effectively trained in early timesteps, manipulating bottleneck features allows for smooth propagation to later denoising steps, ensuring the integrity of complete diffusion semantics is maintained. Moreover, due to the short optimization path, the problem of gradient vanishing is efficiently avoided.

**Noise Maps as Semantic Editors.** Building on our previous analysis, we introduce DragNoise, an interactive point-based image editing method that leverages diffusion semantic propagation. The rationale behind our DragNoise is to treat the predicted noises as sequential semantic editors. Our editing process initiates at a timestep (*e.g.*, t=35) where high-level semantics are well-trained. During this phase, diffusion semantic optimization is conducted on the bottleneck feature of the U-Net to reflect user edits. The optimized bottleneck feature learns the intended dragging effect and produces the corresponding manipulation noise. This optimized bottleneck feature contains target semantics and therefore is propagated

to all subsequent timesteps by substituting the corresponding bottleneck features, avoiding redundant feature optimization. This substitution significantly augments the manipulation effect in a stable and efficient manner. We conducted extensive quantitative and qualitative experiments on the drag-based editing benchmark DragBench [29] and diverse example images to evaluate the efficacy of our DragNoise. Notably, DragNoise significantly cuts down the optimization time by over 50% compared to DragDiffusion. Additionally, we explore the impacts of various initial timesteps on the editing process, the optimization of different layers, and the extent of optimization propagation, which underscore DragNoise's efficiency and flexibility in interactive editing.

## 2. Related Work

**Image Editing.** The field of image editing has advanced remarkably, paralleling improvements in image synthesis quality [7, 14, 15, 26]. Building on StyleGAN [14, 15], numerous studies leverage the latent space of pre-trained GANs for diverse image manipulations, including recoloring [2], semantic or attribute adjustments [24, 25, 28, 31, 37, 40], and style transfer [18, 35, 41].

With the advancement of diffusion models, many researchers have applied diffusion models to similar editing tasks. SDEdit [20] represents an early exploration of semantics in the denoising process, editing images via pixel-level guidance through noise addition and denoising. However, it is primarily limited to global editing. As large-scale pre-trained latent diffusion models (LDMs) [26] emerge as frontrunners in generative modeling, their principles are increasingly utilized in various editing tasks. For localized editing, some studies [9, 10] utilize cross-attention mechanisms. Additionally, given that latent maps are intermediate image representations, they are apt for image editing [6]. However, unlike the latent space in StyleGANs, directly editing the latent maps in diffusion models performs poorly in the presence of significant content or color changes. While most diffusion model editing tasks rely on text prompts [3, 5, 16, 17, 21], there is a growing need for more convenient, precise, and user-friendly interactive control methods for users.

**Point-based Interactive Editing.** Point-based image editing, known for its user-friendliness, enables a wide range of effects from fine-grained adjustments to more extensive transformations. In prior works, this typically involves a two-step optimization in the latent space: motion supervision and point tracking. A notable example utilizing this principle is DragGAN [23]. However, the generative limitations of GANs restrict more complex edits, particularly with real images. FreeDrag [19], another GAN-based approach, adopts a feature-oriented strategy to diminish dependence on point tracking. Yet, it still faces constraints due to GANs' limitations in generative capabilities, inversion efficiency, and the semantic versatility of latent codes. Recently, DragDiffusion [29] has adapted DragGAN's principles to diffusion models, resulting in improved spatial control and greater flexibility. In a similar vein, DragonDiffusion [22] employs classifier guidance and translates editing signals into gradients through feature correspondence loss. Diverging from these methods, our approach exploits diffusion semantics within diffusion models for point-based editing, offering a novel perspective in interactive image manipulation.

## 3. Drag Your Noise

In this section, we present our method, DragNoise, designed for interactive point-based editing. To foster a comprehensive understanding, we begin by revisiting the fundamentals of diffusion models and LDMs. Additionally, we conduct a concise comparative analysis with previous methodologies. Distinguishing DragNoise from earlier approaches, our method focuses on manipulating the predicted noise, encompassing diffusion semantic optimization and diffusion semantic propagation.

### 3.1. Preliminaries

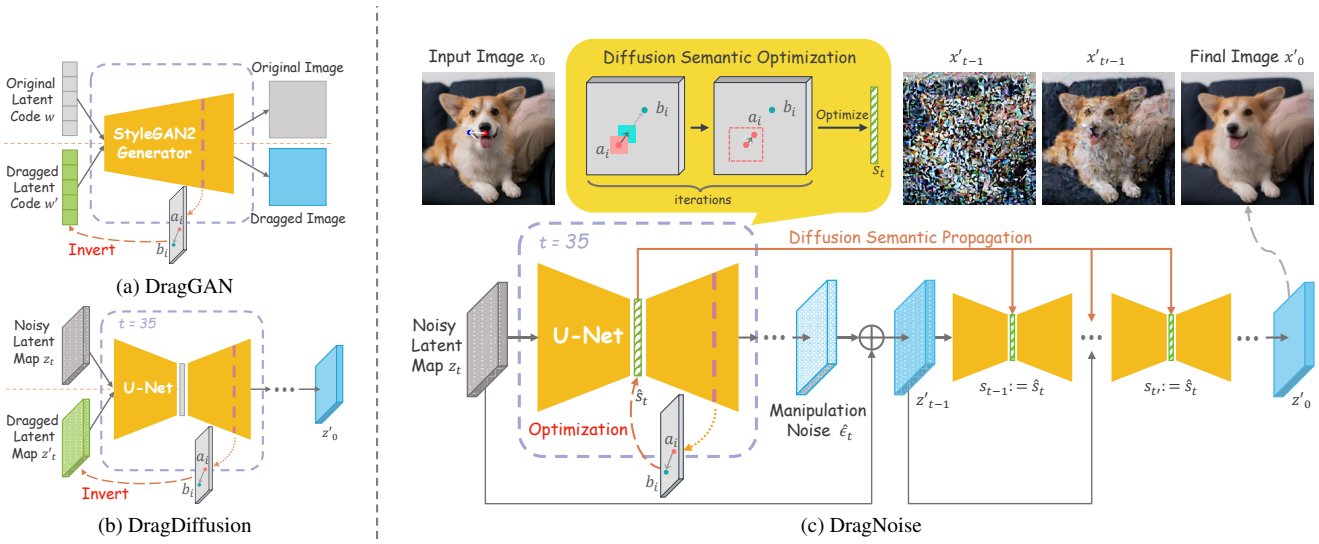
Diffusion models [11, 30] are probabilistic models that learn a data distribution by iteratively denoising noisy images. The forward process refers to a Markov process where the data is corrupted by Gaussian noise. Diffusion models are trained to approximate the reverse process. Practically, a network, typically implemented with U-Net [27], is used to predict the noise $\hat{\epsilon}_t$ at timestep $t$, denoted as $\epsilon_\theta(x_t, t); t = 1, 2, ..., T$. In each timestep, the U-Net receives a noisy image as its input and generates predictions for the noise component at this timestep. Recently, there has been a notable emergence of large-scale diffusion models showing impressive generative capability, especially those based on LDMs, upon which our method is implemented. LDMs compress the denoising process onto the latent space. Given an input noisy latent map $z_t$, the denoising U-Net in LDMs is denoted as $\epsilon_\theta(z_t, t)$.

To enable point-based editing, DragDiffusion extends the inner-inversion method of DragGAN with LDMs (see Figs. 4a and 4b). DragGAN employs feature maps after the 6th block of the StyleGAN2 [15] generator for motion supervision, inverting the editing to the dragged latent code $w'$, which produces the corresponding dragged image. Similarly, DragDiffusion utilizes a U-Net feature map to supervise the optimization of the noisy latent map, yielding a dragged latent map $z'_t$ that governs the subsequent denoising process. Different from their approaches, we manipulate predicted noise and propagate the optimization to edit the image.

### 3.2. Methodology

#### 3.2.1 Diffusion Semantic Optimization

We present an overview of our methodology in Fig. 4c. Based on the analysis in Sec. 1, the bottleneck feature of U-




(a) DragGAN (b) DragDiffusion (c) DragNoise

Figure 4. Comparison between DragNoise and other relevant methods in feature modification, highlighting optimized features in green.

Net in diffusion models has the capability of capturing more comprehensive noise semantics compared to other features. Moreover, the bottleneck feature efficiently captures the majority of semantics at a certain early timestep, denoted as $t$. Therefore, given user-defined point instructions, we perform diffusion semantic optimization with the bottleneck feature at this timestep. Specifically, inspired by Pan *et al*. [23], we perform the optimization process described below.

We denote the user-provided anchor points as $\{a_i = (x_i^a, y_i^a)\}_{i=1,2,...,m}$, and the corresponding objective points as $\{b_i = (x_i^b, y_i^b)\}_{i=1,2,...,m}$, where $m$ is the number of point pairs. Direct spatial mapping between points and the bottleneck feature faces challenges due to significant differences in abstraction levels. To overcome this, we introduce an additional intermediate feature. As demonstrated by Baranchuk et al. [4], the features from the U-Net decoder's third layer are both informative and aware of structure. Therefore, we utilize these features for supervision, achieving an equilibrium between comprehensive structural representation and granularity of detail. Specifically, we obtain a point $p$'s corresponding feature element in this feature map via bilinear interpolation, denoted as $F_p$. To align the structural feature of the anchor point to the objective point, we define the semantic alignment loss as:

$$\mathcal{L}_{alignment} = \sum_{i=1}^{m} \sum_{p_i \in \Omega(a_i, r_1)} \|F_{p_i} - F_{p_i + v_i}\|_1, \quad (1)$$

where $v_i = \frac{b_i - a_i}{\|b_i - a_i\|_2}$ is the normalized vector pointing from $a_i$ to $b_i$ and $\Omega(a_i, r_1) = \{(x, y) | |x - x_i^a| \leqslant r_1, |y - y_i^a| \leqslant r_1\}$ denotes the neighborhood of the anchor point. The semantic alignment loss "drags" the feature of points near the anchor point towards those near the objective point by a small step. Moreover, if a mask is provided, we employ a semantic masking loss to keep the bottleneck feature outside the mask unchanged:

$$\mathcal{L}_{mask} = \|(s_t - \hat{s}_t) \odot (1 - M)\|_1, \quad (2)$$

where $s_t$ is the bottleneck feature at denoising timestep $t$ and $\hat{s}_t$ is its optimized feature. It is noteworthy that, due to the enhanced semantic decoupling of the bottleneck feature, the mask is generally unnecessary for the majority of cases.

In every iteration, we update the bottleneck feature to influence the generation of manipulation noise with those loss terms above. In total, our optimization goal is defined as:

$$\hat{s}_t = \arg\min_{s_t} \ (\mathcal{L}_{alignment} + \lambda \mathcal{L}_{mask}). \quad (3)$$

Note that during back-propagation, the gradient does not propagate backward through $F_{p_i}$.

After each iteration, the bottleneck feature as well as the corresponding feature element of the anchor point are updated. As Eq. (1) no longer provides the accurate direction for optimization iterations afterward, we need to update the location of the anchor point after each iteration. This is achieved by searching in $a_i$'s neighborhood $\Omega(a_i, r_2) = \{(x, y) | |x - x_i^a| \leqslant r_2, |y - y_i^a| \leqslant r_2\}$ for the nearest neighbor of the optimized feature:

$$a_i := \arg\min_{p_i \in \Omega(a_i, r_2)} \|F'_{p_i} - f_i^0\|_1, \quad (4)$$

where $F'_{p_i}$ is feature element of $p_i$ in the updated feature map, and $f_i^0$ denotes the feature element of the initial anchor point.

Finally, diffusion semantic optimization terminates when all the distances between anchor points and their respective objective points are within 1 pixel or when the maximum number of iterations is reached. With the optimized bottleneck feature $\hat{s}_t$, the decoder of the U-Net produces a manipulation noise $\hat{\epsilon}_\theta$.

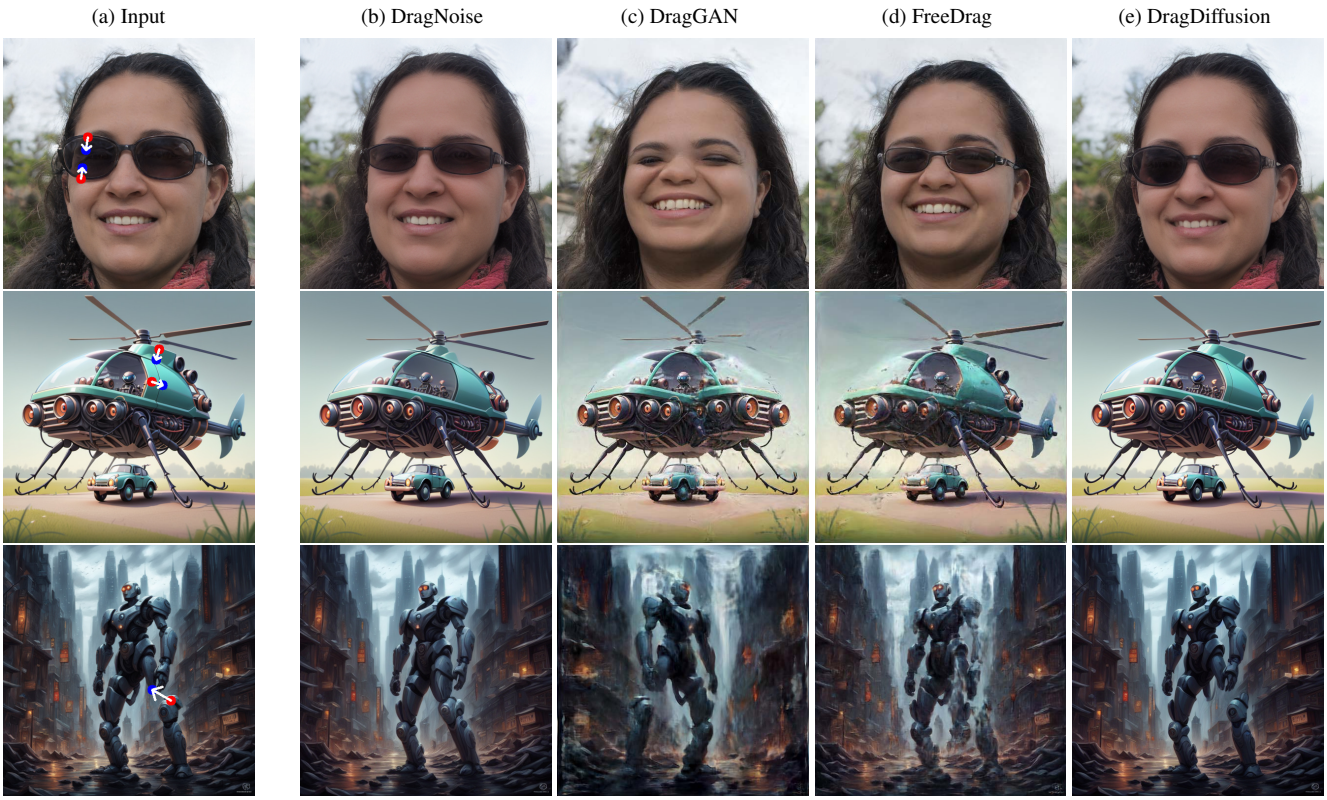



Figure 5. Comparison of point-based editing methods with various drags. Our DragNoise exhibits superior semantic control ability.

#### 3.2.2 Diffusion Semantic Propagation

The manipulation noise $\hat{\epsilon}_\theta$ modifies the denoise direction for the image generation. However, we observe a forgetting issue where subsequent denoising processes tend to overlook the manipulation effect by simply performing diffusion semantic optimization on one timestep (see Sec. 4.3). As we have analyzed in Fig. 2 that propagating the bottleneck feature to later timesteps does not have a significant influence on the overall semantics, we copy this optimized bottleneck feature $\hat{s}_t$ and substitute them in the subsequent timesteps. This is considered as the editing stage that keeps emphasizing the manipulation for the noise prediction. In the final few denoising timesteps, the structure of the image has essentially taken shape. Therefore, we stop replacing the bottleneck feature after timestep $t'$, where we treat it as the refinement stage. After the diffusion semantic propagation in the denoising process, we obtain the generated output latent map $z_0'$ to produce the final image $x_0'$.

### 3.3. Implementation Details

We implement our method based on Stable Diffusion 1.5. Given an input image, to improve the image reconstruction, similar to [29], we train a LoRA [12] with the user-supplied image before diffusion semantic optimization. The fine-tuning process consists of 200 steps to update the LoRA's parameters. Given the need for users to edit anchor points on final generated images and for efficient editing, we treat all images, including diffusion-generated images, as "real images" with classifier-free guidance disabled. We apply DDIM inversion to the given input image to attain the noisy latent map $z_t$ with $t = 35$. With the scheduled total timesteps of 50, during the diffusion semantic optimization stage, the beginning of the editing stage is therefore at $t = 35$ and the refinement stage starts at $t' = 10$. We employ the Adam optimizer with a learning rate of 0.01 for the semantic optimization. The default maximum number of optimization steps is 80. However, it is optional to increase the optimization steps in cases where the user encounters an exceptionally long dragging distance. In Eq. (1) and Eq. (4), we set $r_1$ to 1 and $r_2$ to 3. The value of $\lambda$ in Eq. (3) is set to 0.1.

## 4. Experiments

In this section, we conduct experiments on the drag-based editing benchmark, DragBench [29], as well as diverse example images to assess the effectiveness of DragNoise. We first compare DragNoise with existing GAN-based and diffusion-based methods. Furthermore, we validate the optimization efficiency of our method by comparing it with recent DragDiffusion. Additionally, ablation studies are conducted to analyze the influences of different initial timesteps of the editing stage, the effect of optimization on distinct layers, and the extent of optimization propagation at the editing stage. Quan-

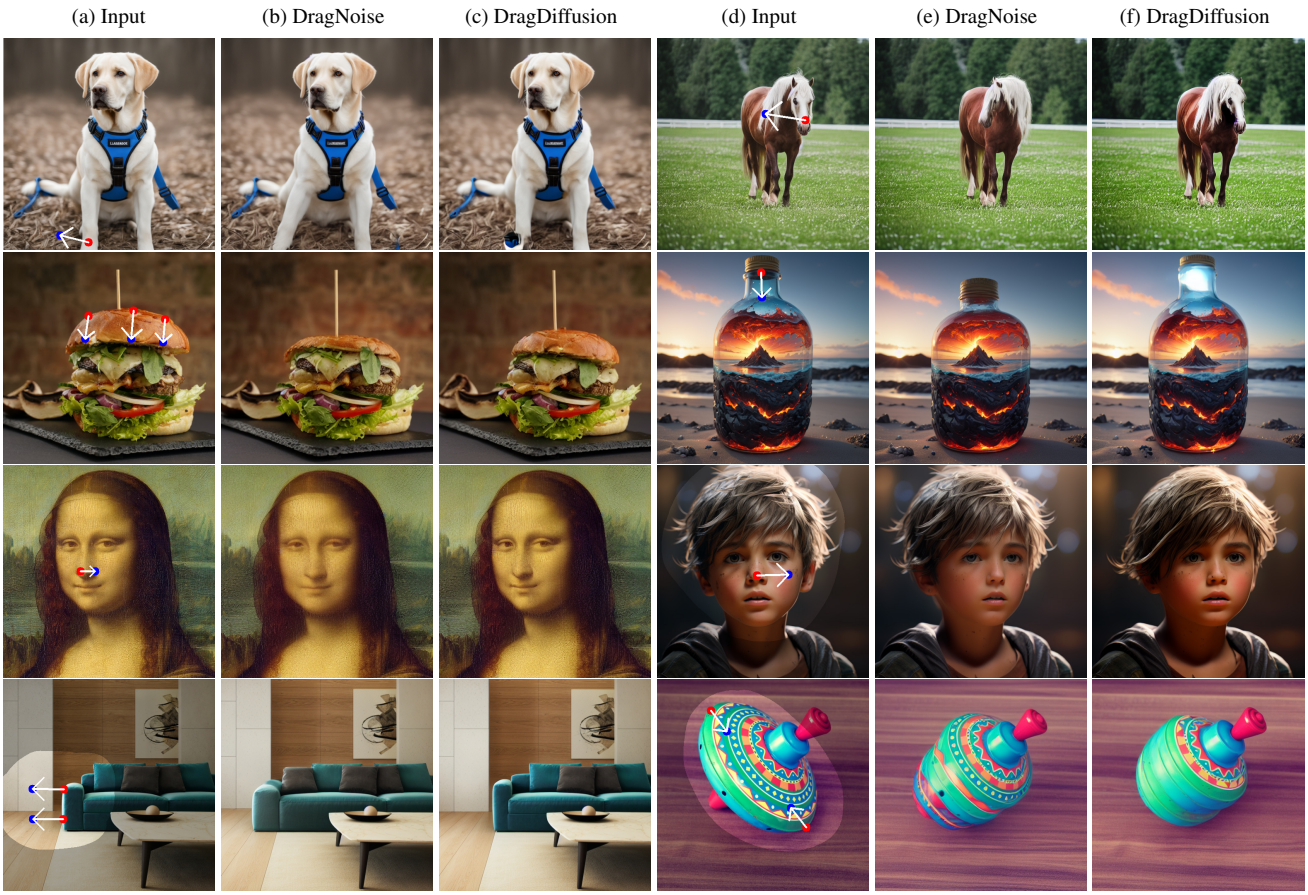



Figure 6. Comparison with DragDiffusion across diverse images. Our method yields more stable and plausible edits aligned with user inputs.

titative experiments using various metrics are carried out to validate our dragging accuracy and image fidelity.

### 4.1. Qualitative Evaluations

In this subsection, we compare our method to existing approaches, *i.e.*, DragGAN [23], FreeDrag [19] and DragDiffusion [29]. For all the input images, we train StyleGAN and LoRA for GAN-based editing and diffusion-based editing, respectively.

In Fig. 5, we conduct a general comparison with GAN-based methods and the diffusion-based method. Both DragGAN and FreeDrag exhibit the capability to relocate anchor points to objective points. However, due to the limited spatial information in their 1D latent space, they usually encounter global changes during the dragging process. For instance, the movement of anchor points results in the unintended shifting of features such as the person's facial shape in the 1st row. Moreover, DragGAN incorrectly tracks new anchor points when the features around the anchor points are similar, leading to unexpected editing effects such as flipping and mirroring (see Fig. 5c). The results show that both GAN-based methods are constrained by the limitations in StyleGAN's generation and inversion capabilities. As a consequence, both DragGAN and FreeDrag yield outcomes where the regions around anchor points become blurred when additional contents are required to be generated to fill the space.

We further showcase diverse results in comparison with DragDiffusion in Fig. 6. Results show that DragDiffusion's two-dimensional noisy latent map offers improved local editing capabilities without inducing global changes. Nevertheless, the inversion capability of the noisy latent map is constrained, resulting in a loss of semantic information. Moreover, the lack of highly decoupled semantic information in the noisy latent map results in semantic alterations in the generated images around the anchor points. For example, as demonstrated in the 3rd row of Fig. 6f, the hairstyle changes after the face rotation. In addition, due to the long chain of back-propagation of DragDiffusion's optimization on the noisy latent map, if the features near the anchor points are similar, *i.e.*, when the semantic alignment loss is small, the gradient vanishes, impeding updates to the noisy latent map and consequently leading to an under-dragged image (*e.g.*, the horse in Fig. 6f).

In comparison, our method precisely moves the anchor points toward the objective points and achieves a natural dragging effect with high fidelity. This is attributed to the proposed diffusion semantic optimization and propagation with the bottleneck feature. For instance, when dragging to

(a) Input (b) Result (c) Input (d) Result

Figure 7. Our DragNoise also supports multi-point editing.

narrow the sunglasses, our modifications do not change the face shape (see the 1st row of Fig. 5b). In both Figs. 5 and 6, our method exhibits superior editing capabilities, ensuring more accurate dragging even amidst substantial changes while also preserving semantics or identities more effectively. Generally, when moving an object, filling a space or resizing an object, changes are confined to the contents associated with the anchor points, preserving the unchanged status of the other semantic contents.

In Fig. 7, we further showcase multi-point control with two examples: precise editing of propellers and legs (Fig. 7b), and lifting a cat's head while keeping its hat in place (Fig. 7d). These cases with multi-points for multi-targets highlight DragNoise's versatility in diverse editing scenarios.

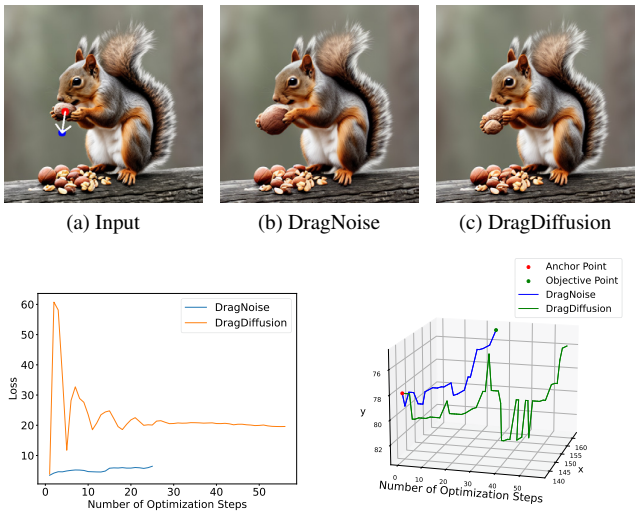
### 4.2. Analysis of Optimization Efficiency

Fig. 8 demonstrates an example of the optimization process of DragNoise and DragDiffusion, depicting their respective changes in loss and the trajectory of the anchor point. DragNoise rapidly determines the direction of semantic editing in the bottleneck feature, approaching the objective point with the optimization process completed in merely 25 steps. In contrast, DragDiffusion requires 56 iterations to reach the objective point.

The optimized bottleneck feature in DragNoise offers a more enriched and higher-level semantic space in contrast to the latent map utilized in DragDiffusion. Consequently, the direction of semantic editing is discernible at the start, leading to a reduction in the number of optimization steps by over 50%. In addition, within a single optimization iteration, DragNoise achieves a 10% reduction in the time required compared to DragDiffusion, attributed to its shorter back-propagation chain. This results in DragNoise cutting down the optimization time by over 50%. On the Tesla V100 graphics card, editing an image with a resolution of $512\times512$ takes approximately 10 seconds for DragNoise, whereas DragDiffusion requires over 22 seconds.

### 4.3. Ablation Study

**Effect of the initial timesteps of the editing stage.** We study the effect of optimizing the bottleneck feature at different beginnings of the editing stage. Fig. 9 shows the effect of the same dragging operation with different initial editing timesteps. The results show that with earlier initial timesteps, *e.g.*, $t = 45$, the bottleneck feature controls the




(a) Input (b) DragNoise (c) DragDiffusion

(d) Optimization Loss (e) Motion Trajectory of Anchor Points

Figure 8. Comparison results of optimization efficiency. (a) and (b) are edited results of the DragNoise and DragDiffusion. (d)) and (e) illustrate the change in total loss and the motion trajectory of the anchor points during the optimization.

larger semantics, shrinking the whole object. With later initial editing timesteps, the smaller the semantics can be controlled, *e.g.*, at $t = 35$ the outer ring of iron was reduced. However, at $t = 30$ only the iron around the anchor points are reduced. This experiment shows the flexibility of our method to control the semantics of various scales.

**Effect of optimizing different features.** We analyze the impact of updating different features from the U-Net when dragging the image while keeping other settings as default. As shown in Fig. 10, optimizing the bottleneck feature yields the optimal results in terms of image quality and controllability, allowing for the most precise control. Optimizing the encoder blocks leads to poor generation quality as the encoder features struggle to capture complete semantics. On the other hand, optimizing decoder blocks faces challenges in determining the direction of semantic editing.

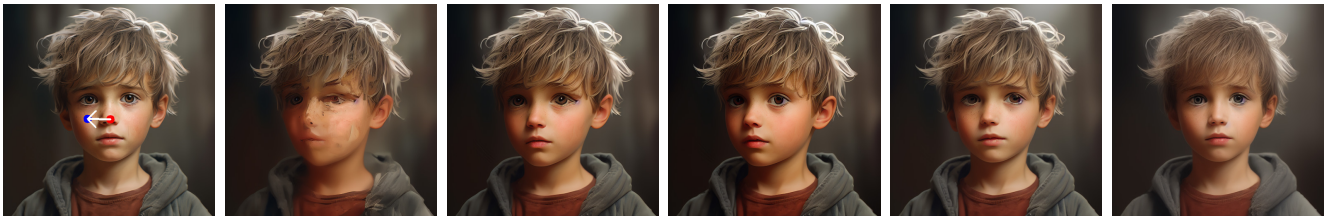
**Effect of different extents of the editing stage.** We analyze the effect of different extents of the editing stage with other settings as default. In Fig. 11, selecting a proper editing endpoint is crucial for image generation quality. Editing until $t' = 20$ leads to instability and vagueness, whereas controlling noise until $t' = 10$ ensures clarity and detail preservation. Beyond $t' = 10$, the last denoising steps reintroduce fine original image details without the optimized bottleneck feature's influence. Conversely, controlling predicted noise throughout can result in the loss of details like teeth or flower textures, yielding unclear outputs.

### 4.4. Quantitative Analysis

We adopt the DragBench dataset and quantitative metrics of mean distance (MD) [23, 29, 33] and image fidelity (IF) [16] for quantitative analyses. IF serves as an indicator of the

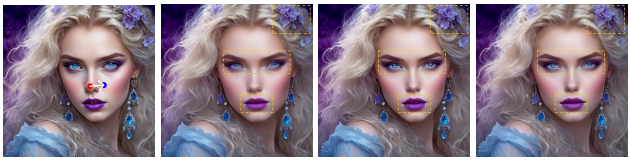
(a) Input (b) t=45 (c) t=40 (d) t=35 (e) t=30

Figure 9. Results of performing the same dragging operation while initiating the editing process with different initial timesteps.



(a) Input (b) Encoder Block 1 (c) Encoder Block 3 (d) Bottleneck (e) Decoder Block 1 (f) Decoder Block 3

Figure 10. Results of different feature layers in U-Net using as optimization targets.



(a) Input (b) Halt at t'=20 (c) Halt at t'=10 (d) No Halting

Figure 11. Copying optimized bottleneck features into the denoising process and halting propagation at different timesteps.

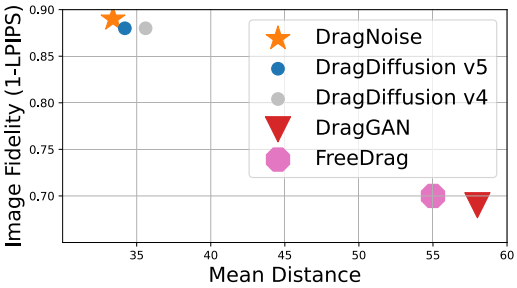



Figure 12. Quantitative evaluations on DragBench, we achieve the best editing accuracy and fidelity.

(a) Input (b) Result (c) Input (d) Result

Figure 13. Failure cases. Edits may negate each other.

fidelity of editing outcomes, and MD reflects the accuracy of the dragging effect. Fig. 12 demonstrates the performance of the four methods on the DragBench dataset, the diffusion-based editing methods generally outperform the GAN-based methods. Notably, our DragNoise, surpasses all existing methods in terms of dragging accuracy and image fidelity.

## 5. Conclusion, Limitation, and Future Work

In this paper, we propose DragNoise, a novel interactive point-based image editing method that leverages diffusion semantic propagation. Drawing upon a thorough exploration of diffusion semantics, we consider the predicted noise in the reverse diffusion process as semantic editors. Our approach involves diffusion semantic optimization and diffusion semantic propagation, enabling the editing of diffusion semantics within a single timestep while effectively propagating the resultant changes. Extensive experiments have validated our method's efficiency and flexibility.

Similar to DragDiffusion, one limitation of our approach is the challenge in handling real images while preserving their original fidelity. Currently, we rely on methods like LoRA to maintain the integrity of real images during the editing process. While effective, this method can sometimes constrain the range and depth of the editing capabilities. Moreover, as shown in Fig. 13, our method sometimes fails in executing intended rotations, underlining the constraints of point-based editing in tasks needing a global perspective, such as novel view synthesis or full object rotation. Future research should aim at creating a universal adapter to maintain diffusion model fidelity and improve editing that requires integrated point understanding.

**Acknowledgement.** The work is supported by the Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2023B1515020097), Singapore MOE Tier 1 Funds (No. MSS23C002), NRF Singapore under the AI Singapore Programme (No. AISG3-GV-2023-011), Guangzhou 2023 Key Research and Development Program on Science and Technology for Agriculture and Social Development (2023B03J1328), and the National Key Technologies Research and Development Program of China (2023YFD2400600).

## References


[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *ICCV*, pages 4432–4441, 2019. 1

[2] Mahmoud Afifi, Marcus A Brubaker, and Michael S Brown. Histogan: Controlling colors of gan-generated and real images via color histograms. In *CVPR*, pages 7941–7950, 2021. 3

[3] Omri Avrahami, Dani Lischinski, and Ohad Fried. Blended diffusion for text-driven editing of natural images. In *CVPR*, pages 18208–18218, 2022. 1, 3

[4] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *ICLR*, 2022. 2, 4

[5] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*, pages 18392–18402, 2023. 1, 3

[6] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *ICCV*, pages 14367–14376, 2021. 3

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021. 1, 2, 3

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NeurIPS*, 27, 2014. 1

[9] Jing Gu, Yilin Wang, Nanxuan Zhao, Tsu-Jui Fu, Wei Xiong, Qing Liu, Zhifei Zhang, HE Zhang, Jianming Zhang, HyunJoon Jung, and Xin Eric Wang. PHOTOSWAP: Personalized subject swapping in images. In *NeurIPS*, 2023. 1, 3

[10] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-or. Prompt-to-prompt image editing with cross-attention control. In *ICLR*, 2022. 1, 3

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020. 1, 3

[12] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *ICLR*, 2021. 5

[13] Yutao Jiang, Yang Zhou, Yuan Liang, Wenxi Liu, Jianbo Jiao, Yuhui Quan, and Shengfeng He. Diffuse3d: Wide-angle 3d photography via bilateral diffusion. In *ICCV*, pages 8998–9008, 2023. 1

[14] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pages 4401–4410, 2019. 3

[15] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, pages 8110–8119, 2020. 1, 3

[16] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *CVPR*, pages 6007–6017, 2023. 1, 3, 7

[17] Dongxu Li, Junnan Li, and Steven Hoi. BLIP-diffusion: Pre-trained subject representation for controllable text-to-image generation and editing. In *NeurIPS*, 2023. 1, 3

[18] Zhansheng Li, Yangyang Xu, Nanxuan Zhao, Yang Zhou, Yongtuo Liu, Dahua Lin, and Shengfeng He. Parsing-conditioned anime translation: A new dataset and method. *ACM TOG*, 42(3):1–14, 2023. 3

[19] Pengyang Ling, Lin Chen, Pan Zhang, Huaian Chen, and Yi Jin. Freedrag: Point tracking is not you need for interactive point-based image editing. *arXiv preprint arXiv:2307.04684*, 2023. 3, 6

[20] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2021. 1, 3

[21] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *CVPR*, pages 6038–6047, 2023. 3

[22] Chong Mou, Xintao Wang, Jiechong Song, Ying Shan, and Jian Zhang. Dragondiffusion: Enabling drag-style manipulation on diffusion models. *arXiv preprint arXiv:2307.02421*, 2023. 3

[23] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *SIGGRAPH*, pages 1–11, 2023. 1, 3, 4, 6, 7

[24] Rishubh Parihar, Ankit Dhiman, and Tejan Karmali. Everything is there in latent space: Attribute editing and attribute style manipulation by stylegan latent space exploration. In *ACM MM*, pages 1828–1836, 2022. 3

[25] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *ICCV*, pages 2085–2094, 2021. 3

[26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 2, 3

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 3

[28] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, pages 9243–9252, 2020. 3

[29] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. *arXiv preprint arXiv:2306.14435*, 2023. 1, 2, 3, 5, 6, 7

[30] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, pages 2256–2265, 2015. 3

[31] Haorui Song, Yong Du, Tianyi Xiang, Junyu Dong, Jing Qin, and Shengfeng He. Editing out-of-domain gan inversion via differential activations. In *ECCV*, pages 1–17, 2022. 3

[32] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2020. 2

[33] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *arXiv preprint arXiv:2306.03881*, 2023. 7

[34] Andrey Voynov, Qinghao Chu, Daniel Cohen-Or, and Kfir Aberman. $p+$: Extended textual conditioning in text-to-image generation. *arXiv preprint arXiv:2303.09522*, 2023. 2

[35] Zongwei Wu, Liangyu Chai, Nanxuan Zhao, Bailin Deng, Yongtuo Liu, Qiang Wen, Junle Wang, and Shengfeng He. Make your own sprites: Aliasing-aware and cell-controllable pixelization. *ACM TOG*, 41(6):1–16, 2022. 3

[36] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *IEEE TPAMI*, 45(3):3121–3138, 2022. 1

[37] Yangyang Xu, Yong Du, Wenpeng Xiao, Xuemiao Xu, and Shengfeng He. From continuity to editability: Inverting gans with consecutive images. In *ICCV*, pages 13910–13918, 2021. 3

[38] Yangyang Xu, Shengfeng He, Kwan-Yee K Wong, and Ping Luo. Rigid: Recurrent gan inversion and editing of real face videos. In *ICCV*, pages 13691–13701, 2023. 1

[39] Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. Paint by example: Exemplar-based image editing with diffusion models. In *CVPR*, pages 18381–18391, 2023. 1

[40] Huiting Yang, Liangyu Chai, Qiang Wen, Shuang Zhao, Zixun Sun, and Shengfeng He. Discovering interpretable latent space directions of gans beyond binary attributes. In *CVPR*, pages 12177–12185, 2021. 3

[41] Shuai Yang, Liming Jiang, Ziwei Liu, and Chen Change Loy. Pastiche master: Exemplar-based high-resolution portrait style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7693–7702, 2022. 3

[42] Yuyang Yu, Bangzhen Liu, Chenxi Zheng, Xuemiao Xu, Huaidong Zhang, and Shengfeng He. Beyond textual constraints: Learning novel diffusion conditions with fewer examples. In *CVPR*, 2024. 1

[43] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pages 586–595, 2018. 2

[44] Chenxi Zheng, Bangzhen Liu, Xuemiao Xu, Huaidong Zhang, and Shengfeng He. Learning an interpretable stylized subspace for 3d-aware animatable artforms. *IEEE TVCG*, 2024. 1

[45] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*, pages 592–608, 2020. 1