DRAGTEXT: Rethinking Text Embedding in Point-based Image Editing
– Supplementary Material –

Contents

A. Additional Material: Code & Project Page

The code has been submitted as a zip file along with the Supplement. Our results are presented in an easily accessible format on our project page. The link to the project page is as follows: `https://micv-yonsei.github.io/dragtext2025/`

B. More Details on Prompt Engineering

In this section, we provide a detailed explanation of the analysis from Section 3.2., examining the effectiveness of prompt engineering in point-based image editing. First, we explain how the analysis was conducted. Next, we present more examples of the intention text we used and the corresponding results.

B.1. Implementation Details

Drag Editing with Intention Text. We estimated the editing intentions for each image using the handle points, the target points, and the image masks provided by the DragBench dataset [8]. Additionally, we referenced the edited results from four methods [3, 4, 8, 12]. The intention text prompts were crafted by injecting these editing intentions into the original text prompts. To ensure that secondary changes in the text prompts did not affect the editing results, we minimized alterations to the vocabulary and sentence structures of the original text prompts.

For example, consider an image with the original prompt `"a photo of a jug and a glass"` where the jug's neck needs to be shortened. We can craft an intention text prompt via [5] such as:

```
"Create an image of a jug with a shorter neck.  Shorten the neck by the distance
between a red dot and a blue dot.  The jug should have a smooth, glossy finish.
Place the jug against a simple, neutral background."
```

However, this significantly altered the content of the original text prompt. This alteration makes it challenging to discern whether the changes in the edited result were due to these secondary modifications or the incorporation of the editing intention in the text. Consequently, we incorporated concise terms representing the editing intention while preserving the original vocabulary and sentence structure as much as possible, for example: `"a photo of a` *`short-neck`* `jug and a glass."`

Linear Interpolation. To reflect gradual changes in the image embeddings to the text embeddings, we linearly interpolate between the original text embeddings and the intention text embeddings during the dragging process. The weights of the original text embeddings and the intention text embeddings are determined based on the distance between the handle point h_i^k and the target point g_i:

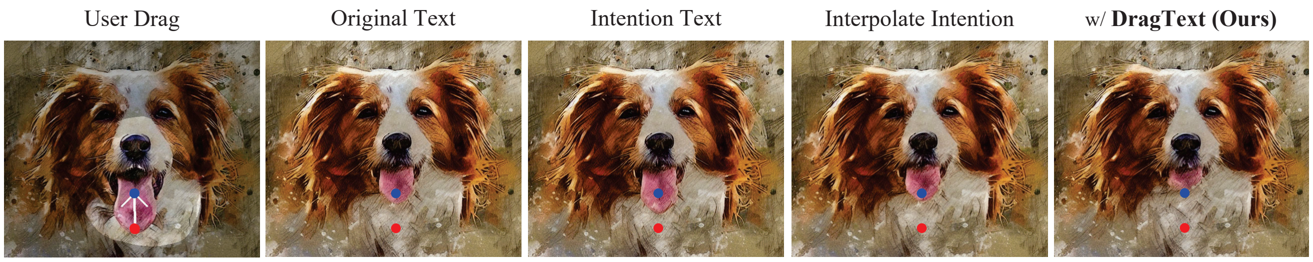
$$w^k = \frac{\sum_{i=1}^{n} \|g_i - h_i^k\|_2}{\sum_{i=1}^{n} \|g_i - h_i^0\|_2}$$

$$\hat{\mathbf{c}}^k = w^k \cdot \hat{\mathbf{c}}^k_{\text{org}} + (1 - w^k) \cdot \hat{\mathbf{c}}^k_{\text{int}}$$

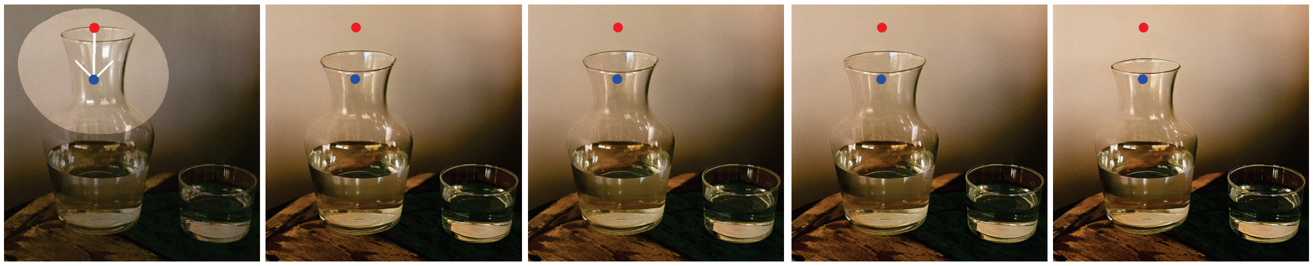
where $\hat{\mathbf{c}}^k_{\text{org}}$ is the original text embedding and $\hat{\mathbf{c}}^k_{\text{int}}$ is the intention text embedding. At the beginning of the point-based editing, $w^k = 1$ so $\hat{\mathbf{c}}^k = \hat{\mathbf{c}}^k_{\text{org}}$. Conversely, as the dragging progresses and handle points approach target points, the weight value w^k increases, resulting in a higher proportion of $\hat{\mathbf{c}}^k_{\text{int}}$. In this way, as the image is progressively edited, the proportion of the intention text embedding is gradually increased.

B.2. More Qualitative Results for Prompt Engineering

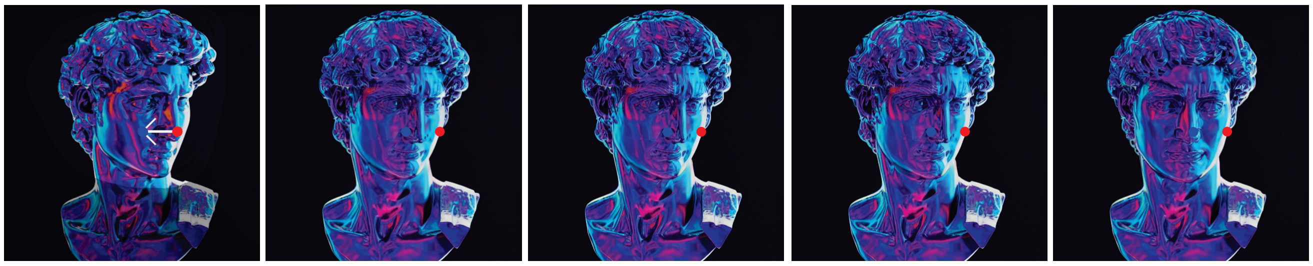
In Fig. 1, we present additional results for prompt engineering. Corresponding results for DRAGTEXT are also presented to validate the effectiveness of our approach in comparison to prompt engineering. Prompt engineering was found to have little impact on alleviating drag halting. In contrast, DRAGTEXT dragged handle points closer to target points, compared to the original text prompt, the intention text prompt, and their interpolation.



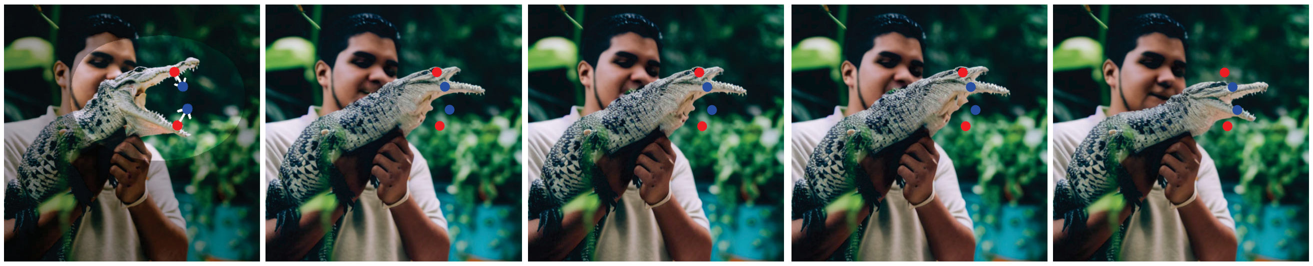
(a) "The oil painting of a dog with its mouth closed“



(b) "a photo of a short neck jug and a glass“



(c) “A sculpture of a man head facing forward”



(d) “A photo of a man holding a crocodile with its mouth closed”

Figure 1. **More qualitative results for prompt engineering (*i.e.* the intention text, and interpolated intention text).** Red points and blue points represent handle points and their target points respectively. Overall, DRAGTEXT moved the content at the handle point of the User Drag image closer to the target point. Additionally, in (b), it was observed that interpolating the intention text with the original text makes it challenging to maintain the object’s style. In (d), DRAGTEXT preserved semantics better than other methods while dragging appropriately.

C. More Details on DRAGTEXT

In this section, we provide a comprehensive overview of our method to ensure clarity and ease of understanding. We describe the pipeline of DRAGTEXT using pseudo-code to aid in understanding our approach. Additionally, we detail the modifications necessary to apply DRAGTEXT to other point-based image editing methods.

C.1. Pseudo Code of DRAGTEXT

Algorithm 1 Pipeline of DRAGTEXT

Input: Input image $\mathbf{x}_0$, text prompt $\mathbf{c}$, image mask M_{image}, handle points $\{h_i\}_{i=1}^n$, target points $\{g_i\}_{i=1}^n$, denoising U-Net U_θ, diffusion time step t, maximum number of iterations steps K
Output: Output image $\hat{\mathbf{x}}_0$

```
1:  $\mathbf{z}_t \leftarrow$ apply DDIM inversion to $\mathbf{x}_0$ conditioned on $\mathbf{c}$
2:  $\mathbf{z}_t^0, \mathbf{c}^0, h_i^0 \leftarrow \mathbf{z}_t, \mathbf{c}, h_i$
3:  for $k$ in $0 : K-1$ do
4:      $\mathcal{F}(\hat{\mathbf{z}}_t^k, \hat{\mathbf{c}}^k) \leftarrow U_\theta(\hat{\mathbf{z}}_t^k; \hat{\mathbf{c}}^k)$
5:      Update $\hat{\mathbf{z}}_t^k$ using motion supervision
6:      Update $\hat{\mathbf{c}}^k$ using text optimization
7:      $\hat{\mathbf{z}}_t^{k+1}, \hat{\mathbf{c}}^{k+1} \leftarrow \hat{\mathbf{z}}_t^k, \hat{\mathbf{c}}^k$
8:      Update $\{h_i^{k+1}\}_{i=1}^n$ using points tracking
9:  $\hat{\mathbf{z}}_t, \hat{\mathbf{c}} \leftarrow \hat{\mathbf{z}}_t^K, \hat{\mathbf{c}}^K$
10: for $t$ in $t : 1$ do
11:     $\hat{\mathbf{z}}_{t-1} \leftarrow$ apply denoising to $\hat{\mathbf{z}}_t$ conditioned on $\hat{\mathbf{c}}$
12: $\hat{\mathbf{x}}_0 \leftarrow \hat{\mathbf{z}}_0$
```

In DragText, another important element is the mask $M_{\text{text}} \in \mathbb{R}^{l \times d}$. This mask is used to regularize text embedding optimization but is not an input. Instead, it is automatically calculated by the CLIP tokenizer [7]. After the text prompt passes through the tokenizer, the tokens excluding `[EOS]` and `[PAD]` tokens contain significant semantic information. The length of these important tokens is l.

C.2. Modifications for Integrate with Other Methods

In the main paper, we explained DRAGTEXT based on DragDiffusion [8]. We chose this method because representative diffusion model-based dragging methods [3, 4, 12] all utilize approaches from DragGAN [6] and DragDiffusion. Therefore, they are constructed upon the foundation of DragDiffusion. However, they also developed techniques to overcome the limitations of DragDiffusion. Taking these improvements into account, we made minor modifications to DRAGTEXT to adapt our approach for each method.

C.2.1 FreeDrag

Point-based image editing has faced challenges, such as the disappearance of handle points during the dragging process. Additionally, point tracking often fails to reach the target point because it moves to the location with the smallest difference in the feature map. To address these issues, FreeDrag introduces *Template Feature* and restricts the path of point tracking to a straight line.

FreeDrag generates the corresponding template features $\mathcal{T}_i^k$ for each of the n handle points. During the optimization step, the feature map is optimized to match the template features:

$$\begin{aligned}\mathcal{L}_{\text{ms}} = &\sum_{i=1}^{n} \sum_{q_1} \left\| \mathcal{F}_{q_1+d_i}(\hat{\mathbf{z}}_t^k, \hat{\mathbf{c}}^k) - \mathcal{T}_i^k \right\|_1 \\ &+ \lambda_{\text{image}} \left\| \left(\hat{\mathbf{z}}_{t-1}^k - \text{sg}(\hat{\mathbf{z}}_{t-1}^0)\right) \odot (\nVdash - M_{\text{image}}) \right\|_1 .\end{aligned}$$

This involves up to five iterations of motion supervision until the predefined conditions are met. Depending on the outcome, feature adaptation is categorized into (a) well-learned features, (b) features in the process of learning, and (c) poorly learned features. Point tracking is then performed based on these categories. This process is repeated until the handle points reach the target points, after which the image is denoised to produce the edited image.

In FreeDrag, if the template feature is poorly learned (category (c)), the point not only reverts to its previous position but also reuses the template feature map without updating its values. Inspired by this approach, our DragText computes the text loss only during the (a) and (b) processes to align with the image. In cases categorized as (c), the text embedding is excluded from the optimization process. Therefore, we define α_i as 1 for cases (1) and (2), and 0 for case (3). The text loss is then defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{text}} = &\sum_{i=1}^{n}\sum_{q_1} \alpha_i \left\| \mathcal{F}_{q_1+d_i}(\hat{\mathbf{z}}_t^k, \hat{\mathbf{c}}^k) - \mathcal{T}_i^k \right\|_1 \\ &+ \lambda_{\text{text}} \left\| \left(\hat{\mathbf{c}}^k - \text{sg}(\mathbf{c}^0)\right) \odot M_{\text{text}} \right\|_1 .\end{aligned}$$

In DragText, during image optimization, $\hat{\mathbf{c}}^k$ does not undergo gradient descent, and during text optimization, the latent vector $\hat{\mathbf{z}}_t^k$does not undergo gradient descent.

C.2.2 DragNoise

The bottleneck features $\mathbf{s}_t$ effectively capture richer noise semantics and efficiently capture most semantics at an early timestep t. Thus, DragNoise optimizes the bottleneck feature $\mathbf{s}_t$ of the U-Net instead of the latent vector $\mathbf{z}_t$ thereby shortening the back-propagation chain. Accordingly, DRAGTEXT optimizes $\mathbf{s}_t$ instead of $\mathbf{z}_t$ during the image optimization processes. For each iteration k, $\hat{\mathbf{s}}_t^k$ undergoes a gradient descent step to minimize $\mathcal{L}_{\text{ms}}$:

$$\hat{\mathbf{s}}_t^{k+1} = \hat{\mathbf{s}}_t^k - \eta_{\text{ms}} \frac{\partial \mathcal{L}_{\text{ms}}(\hat{\mathbf{s}}_t^k, \hat{\mathbf{c}}^k)}{\partial \hat{\mathbf{s}}_t^k}.$$

In DRAGTEXT, neither the latent vector $\mathbf{z}_t^k$ nor the bottleneck feature $\mathbf{s}_t^k$ undergoes gradient descent during the text optimization. So the text optimization procedure is not modified in DragNoise.

C.2.3 GoodDrag

GoodDrag alternates between dragging and denoising, introducing periodic corrections to mitigate accumulated errors. This approach is different from traditional diffusion-based dragging methods. They generally execute all drag operations at once before denoising the optimized noisy latent vector $\hat{\mathbf{z}}_t$. During the denoising process, which involves sampling images $\hat{\mathbf{x}}_0$ from a noisy latent vector $\hat{\mathbf{z}}_t$, perturbations from dragging are corrected. However, if the denoising process is performed only after all drag operations are completed, the errors accumulate too significantly to be corrected with high fidelity. To address this, GoodDrag applies one denoising operation after B image optimization and point tracking steps.

For example, the latent vector $\hat{\mathbf{z}}_t^k$ has been denoised $\lfloor \frac{k}{B} \rfloor$ times, the drag optimization is performed at the timestep $t = T - \frac{k}{B}$. To ensure this process is consistent, the total number of drag steps K should be divisible by B. Since DRAGTEXT performs one text optimization step after one image optimization step, we sequentially repeat the image optimization, text optimization, and point tracking steps B times, and then apply one denoising operation.

Moreover, when drag editing moves the handle points $\{h_i\}_{i=1}^n$, the features around handle points tend to deviate from their original appearance. This deviation can lead to artifacts in the edited images and difficulties in accurately moving the handle points. To prevent this, GoodDrag keeps the handle point h_i^k consistent with the original point h_i^0, throughout the entire editing process:

$$\begin{aligned}\mathcal{L}_{\text{ms}} = &\sum_{i=1}^{n}\sum_{q_1} \left\| \mathcal{F}_{q_1+4\cdot d_i}(\hat{\mathbf{z}}_t^k, \hat{\mathbf{c}}^k) - \text{sg}(\mathcal{F}_{q_1^0}(\hat{\mathbf{z}}_t^0, \hat{\mathbf{c}}^0)) \right\|_1 \\ &+ \lambda_{\text{image}} \left\| \left(\hat{\mathbf{z}}_{t-1}^k - \text{sg}(\hat{\mathbf{z}}_{t-1}^0)\right) \odot (\mathbb{1} - M_{\text{image}}) \right\|_1 ,\end{aligned}$$

where $q_1 = \Omega(h_i^k, r_1)$, and q_1^0 describes the square region centered at the original handle point h_i^0. And, drag operations per denoising step $B = 10$. Similarly, DRAGTEXT ensures the handle point h_i^k remains consistent with the original point h_i^0

during text optimization:

$$\mathcal{L}_{\text{text}} = \sum_{i=1}^{n} \sum_{q_1} \left\| \mathcal{F}_{q_1+4\cdot d_i}(\hat{\mathbf{z}}_t^k, \hat{\mathbf{c}}^k) - \text{sg}(\mathcal{F}_{q_1^0}(\hat{\mathbf{z}}_t^0, \hat{\mathbf{c}}^0)) \right\|_1 + \lambda_{\text{text}} \left\| \left(\hat{\mathbf{c}}^k - \text{sg}(\mathbf{c}^0) \right) \odot M_{\text{text}} \right\|_1 .$$

Additionally, GoodDrag faced increased optimization difficulty from this design, due to the larger feature distance compared to the original motion supervision loss. To mitigate this, a smaller step size and more motion supervision steps are used for optimization. This strategy is also applied in DRAGTEXT.

D. Implementation Details

Table 1. Hyperparameters for point-based image editing methods and DRAGTEXT.

Methods	DragDiffusion	FreeDrag	DragNoise	GoodDrag
Diffusion Model	Stable Diffusion 1.5			
Time Step (T)	35	35	35	38
LoRA Training Step	80	200	200	70
Maximum Optimization Step (K)	80	300	80	70
Square radius r_1	1	3	1	4
Motion Supervision Loss				
Learning Rate (η_{ms})	0.01	0.01	0.02	0.02
Optimizer	Adam			
λ_{image}	0.1	10	0.2	0.2
+ Text Optimization Loss (DRAGTEXT)				
Learning Rate η_{text}	0.004			
Optimizer	Adam			
λ_{text}	0.1			
Point Tracking				
Square radius r_2	3	-	3	12
Drag Optimization per Point Tracking	1	-	1	3

In Table 1, we listed the hyperparameters used for each point-based image editing method [3, 4, 8, 12]. These values were consistently used in both the *Baseline* and *w/* DRAGTEXT experiments. For a fair comparison, we applied the same hyperparameter values from the respective paper to our experiments. Additionally, we maintained the same text optimization loss across all methods to demonstrate the robustness of our approach.

In FreeDrag, values related to point tracking are omitted since it replaces point tracking with line search.

E. More Qualitative Results

In Fig. 2, we additionally present the results of applying DRAGTEXT to each method [3, 4, 8, 12]. In our experiments, we applied DRAGTEXT to various point-based image editing methods and evaluated their performance. The results show that DRAGTEXT can effectively drag the handle points to their corresponding target points while maintaining the semantic integrity of the original image. Moreover, the consistent success of DRAGTEXT across multiple methods underscores its robustness and adaptability.

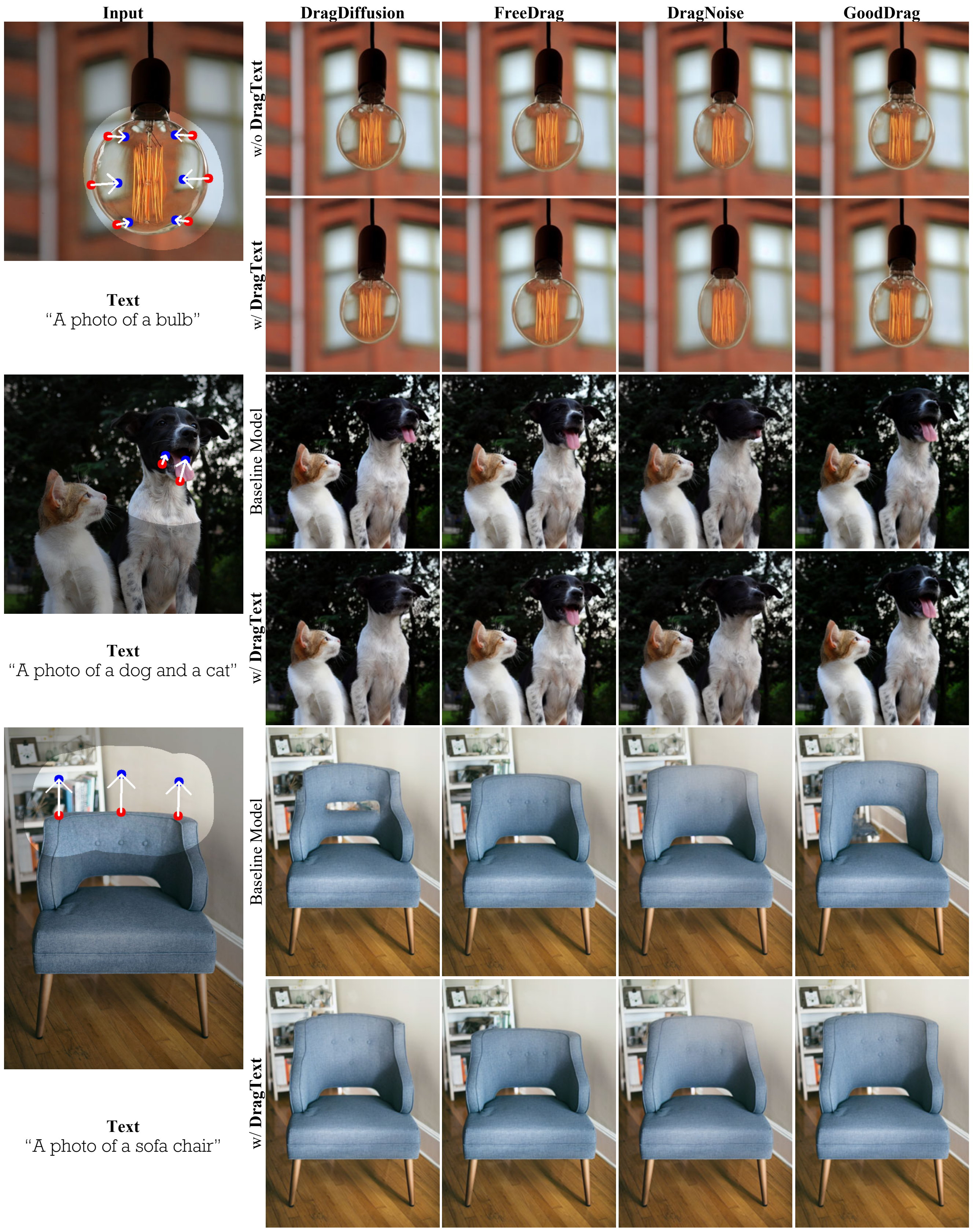


Figure 2. **DRAGTEXT effectively moved the handle point to the target point while preserving semantics.** Moreover, the consistent performance across different methods demonstrates the generalizability of DRAGTEXT.

F. Evaluation Metrics

F.1. LPIPS

LPIPS [11] uses ImageNet classification models such as VGG [9], SqueezeNet [1], and AlexNet [2]. We measured LPIPS using AlexNet. LPIPS measures the similarity between two images by calculating the Euclidean distance of the activation maps obtained from several layers of a pre-trained network, scaling them by weights w, and then averaging the values channel-wise to compute the final LPIPS score.

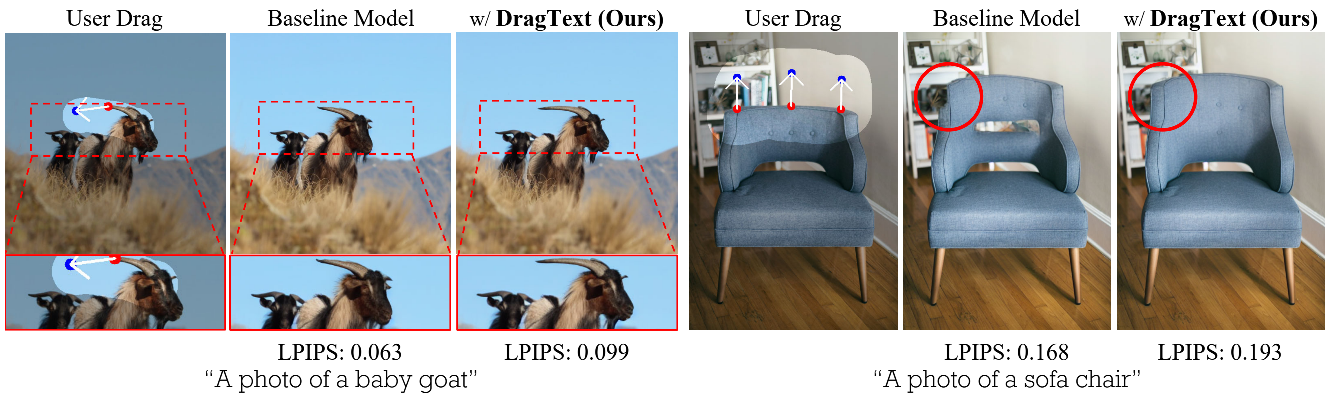


Figure 3. **Comparing the LPIPS scores of the baseline model and DRAGTEXT.** Despite DRAGTEXT achieving better edits and preserving semantics well, its LPIPS scores are worse. In the left image, the baby goat's face direction in the input image is the same in the result with DRAGTEXT. In contrast, the baseline model has the baby goat looking slightly more forward. In the right image, unlike the baseline model, DRAGTEXT maintains the complete form of the sofa chair.

LPIPS is an appropriate metric for measuring the similarity between two images, emphasizing that image editing should maintain similarity to the original image. However, due to the nature of the drag editing task, the image will inevitably change. Consequently, even when dragging is performed successfully, the LPIPS score might worsen. For instance, if an image does not change at all, it would yield an LPIPS score of 0, the best possible score. As shown in Fig. 3, even though we achieved a more desirable image editing outcome, the LPIPS score was lower. Therefore, we propose that LPIPS should not be overly emphasized if the score falls below a certain threshold. To address this issue, we suggest using the product of LPIPS and MD, which are complementary metrics, as a more robust evaluation metric.

F.2. Mean Distance

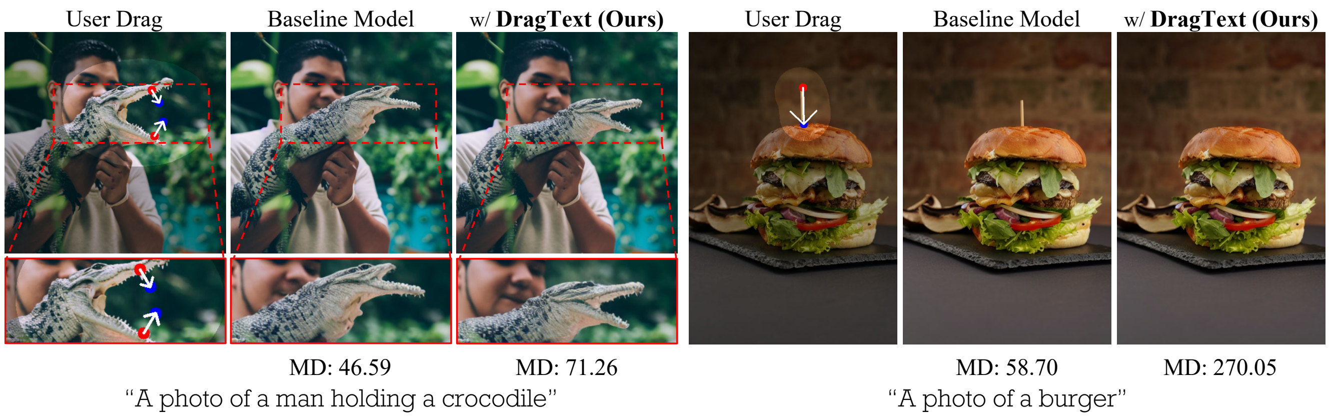


Figure 4. **Comparing the MD scores of the baseline model and DRAGTEXT.** Despite DRAGTEXT moving the handle point closer to the target point, the MD score was lower. In the left image, the baseline model showed a better MD score, despite the crocodile's snout shape being distorted compared to DRAGTEXT. In the right image, DRAGTEXT completed dragging, but its MD score is worse.

Mean distance (MD) is computed via DIFT [10]. First, DIFT identifies corresponding points in the edited image that correspond to the handle points in the original image. These identified points are regarded as the final handle points after editing is complete. Then, the mean Euclidean distance between the corresponding point and the target point is calculated. MD is the average value of all handle-target point pairs.

We propose that evaluating drag editing using Mean Distance (MD) on certain images in the DragBench dataset is challenging. Some images in DragBench require specific objects to disappear through drag editing as the points move. However, if a specific object disappears, there would be no corresponding objects in the edited image, resulting in a significantly high MD value. For instance, in Fig. 4, the handle point and target point indicate that the toothpick should be perfectly inserted into the hamburger. Despite successfully achieving this, DIFT fails to recognize the toothpick, resulting in a higher MD value being calculated. Conversely, there are cases where the MD value is low because the points remain in the same semantic position, but the actual image editing was unsuccessful due to distorted shapes and loss of semantics. While MD is an excellent metric for tasks involving moving feature points of objects, it has certain limitations and challenges when applied to all images in point-based editing tasks.

G. Visual Ablation on the Hyperparameters of Regularization

In Fig. 5, we provide extra visual ablation results to demonstrate how the hyperparameter λ_{text} impacts the regularization process in text optimization. We modified images by adjusting λ_{text} within a range from 0 to 10, which allowed us to control the level of regularization applied during the text optimization phase. When λ_{text} is close to 0, it results in some of the important semantic information being lost. On the other hand, applying an excessively high λ_{text}, prevents the optimization of the text embedding from effectively altering the image.

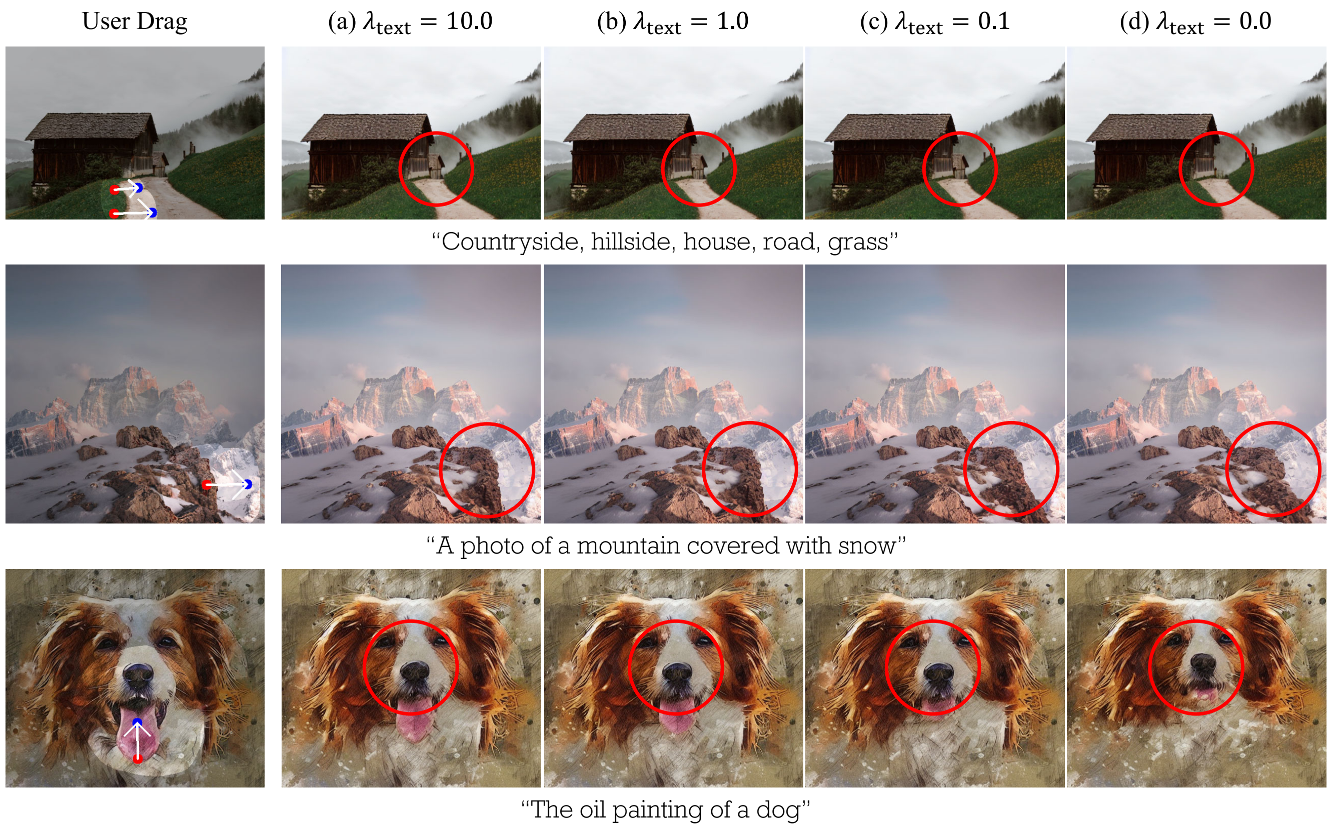

User Drag
(a) $\lambda_{\text{text}} = 10.0$
(b) $\lambda_{\text{text}} = 1.0$
(c) $\lambda_{\text{text}} = 0.1$
(d) $\lambda_{\text{text}} = 0.0$
"Countryside, hillside, house, road, grass"
"A photo of a mountain covered with snow"
"The oil painting of a dog"

Figure 5. **Effect of text regularization with visual ablations on the hyperparameter λ_{text}.** The red circles indicate the loss of semantics. In the first column, the small house on the right has disappeared. In the second column, the cliff's shape has collapsed. In the third column, the position of the dog's facial features has changed.

H. Ablation on the U-Net Feature Maps

We utilize various U-Net decoder blocks for DRAGTEXT with the image embedding fixed from the 3rd block. In Fig. 6 and Table 2, The 3rd block maintains semantics and achieves effective dragging. Lower blocks (*e.g.*, Block 1) have difficulty with semantics, and higher blocks (*e.g.*, Block 4) exhibit poor dragging.

Table 2. Quantitative evaluation results by U-Net block number applied in DRAGTEXT

U-Net Block #	LPIPS↓	MD↓
Block #1	0.135	36.02
Block #2	0.154	37.08
Block #3	0.124	**31.96**
Block #4	**0.119**	33.60

I. More Qualitative Results for Manipulating Embeddings

In Fig. 7 and Fig. 8, we apply linear interpolation and extrapolation to the image and text embeddings to generate not only the intermediate stages of the image editing process but also the parts beyond the editing process. This is possible because DRAGTEXT optimizes both the text and image embeddings simultaneously.

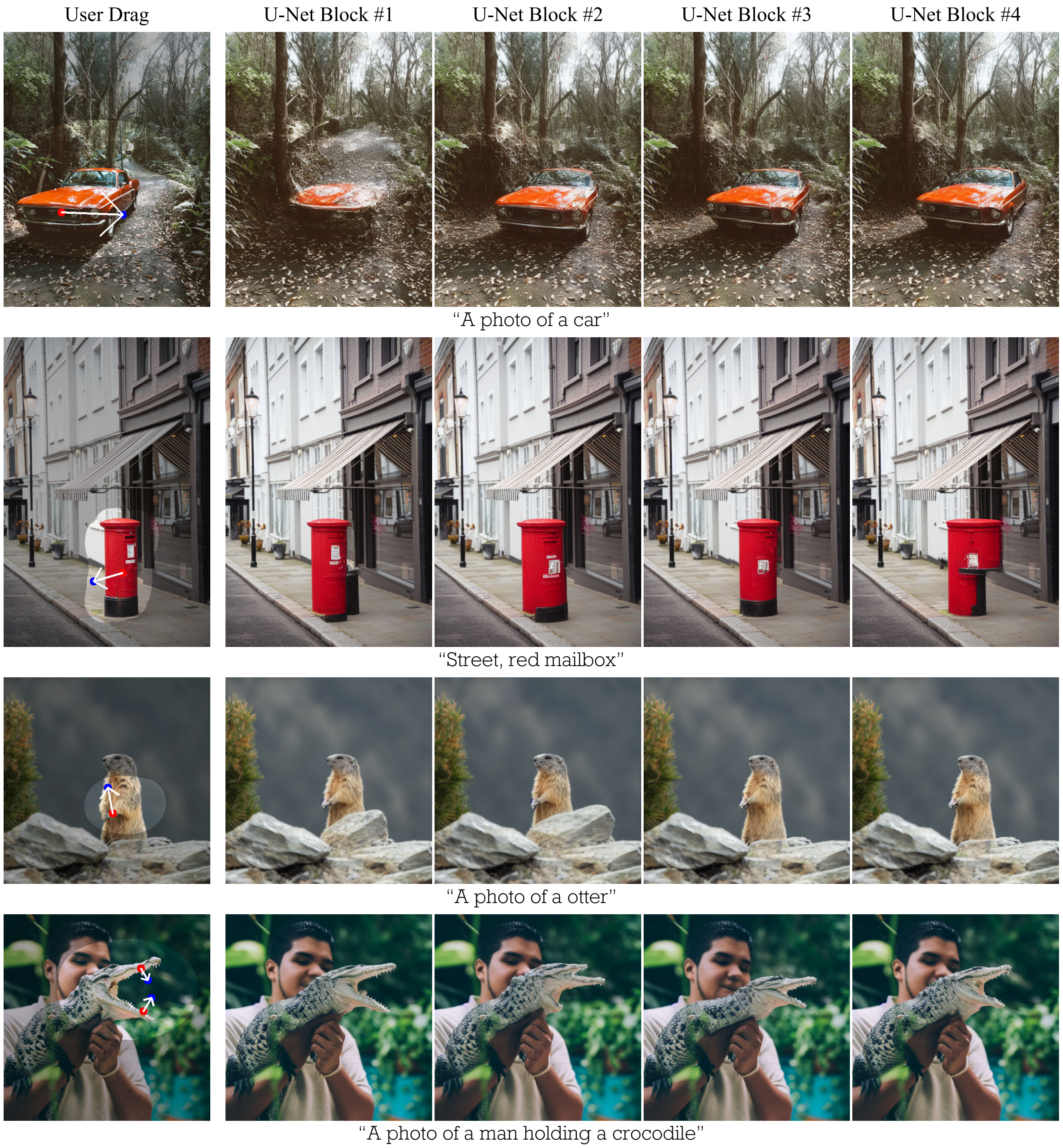


Figure 6. **More image generation results per U-Net decoder block.** Optimizing the text embedding using the 3rd block of the U-Net decoder yields the best performance in terms of dragging and image semantics.

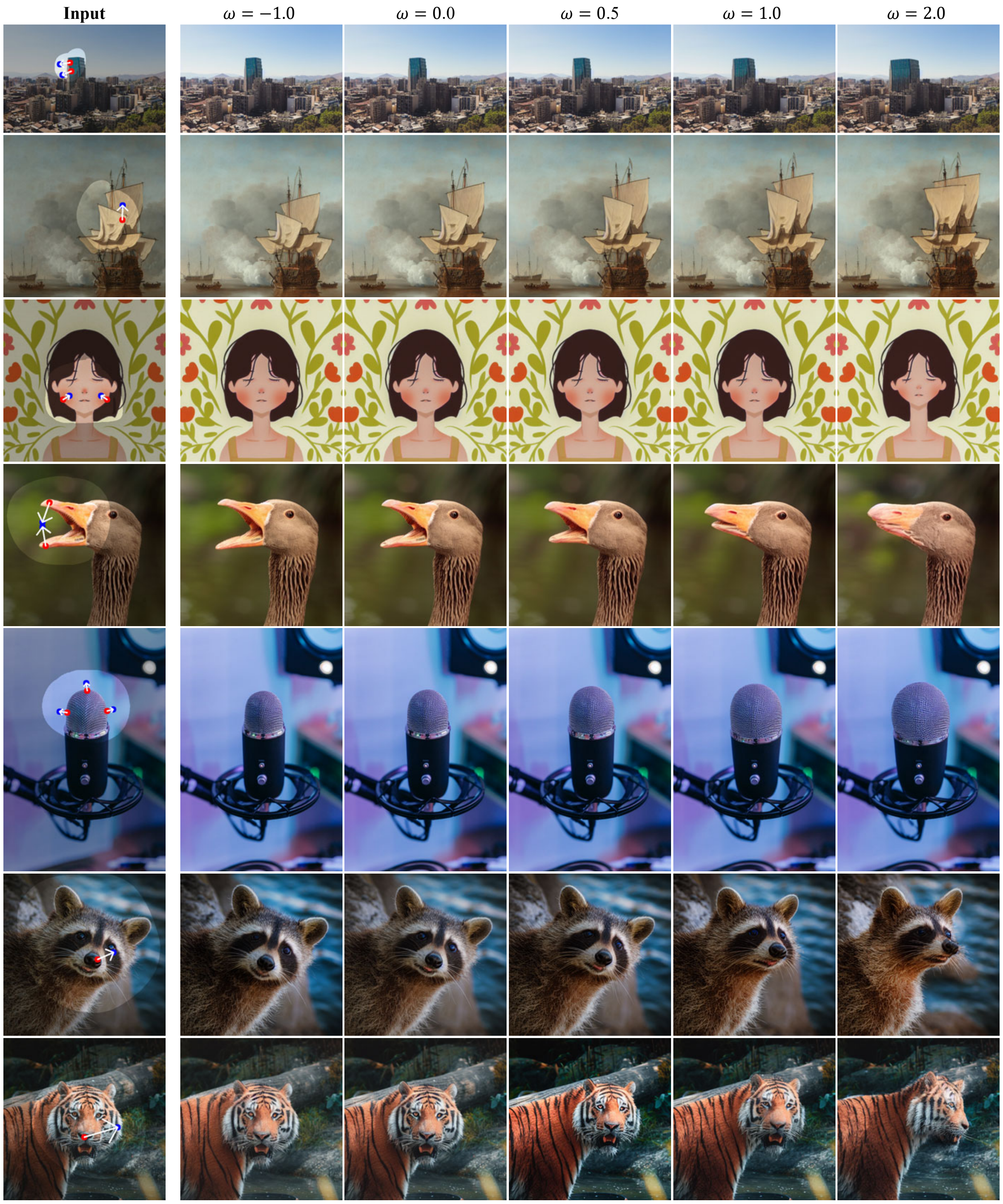


Figure 7. **More qualitative results for manipulating embeddings with DRAGTEXT.** DragText can generate images outside the originally intended editing range in the direction of the drag while preserving the semantic content at the same time.

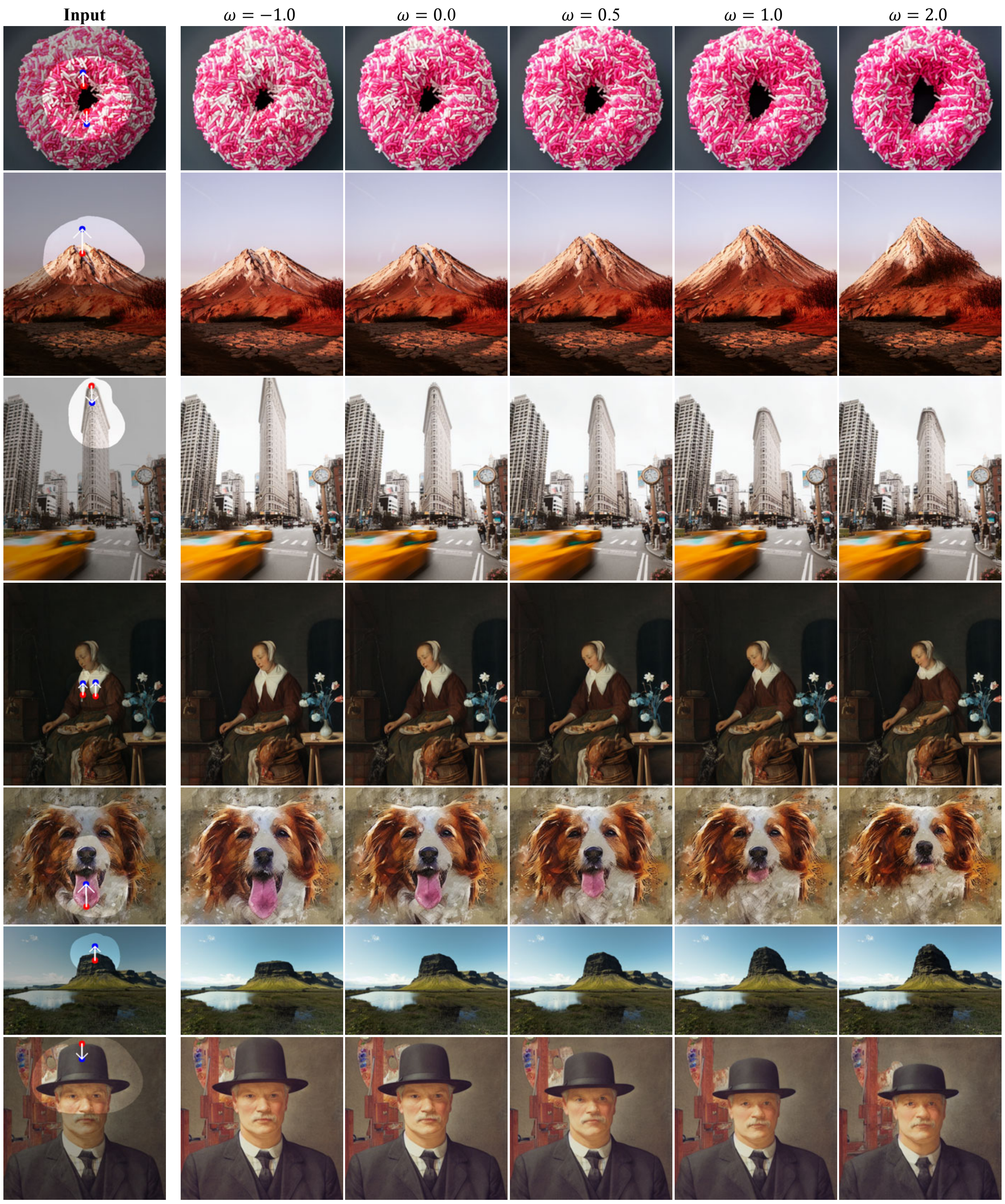


Figure 8. **More qualitative results for manipulating embeddings with DRAGTEXT**. DragText can generate images outside the originally intended editing range in the direction of the drag while preserving the semantic content at the same time.

References

[1] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 8

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 8

[3] Pengyang Ling, Lin Chen, Pan Zhang, Huaian Chen, Yi Jin, and Jinjin Zheng. Freedrag: Feature dragging for reliable point-based image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6860–6870, 2024. 2, 4, 6

[4] Haofeng Liu, Chenshu Xu, Yifei Yang, Lihua Zeng, and Shengfeng He. Drag your noise: Interactive point-based editing via diffusion semantic propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6743–6752, 2024. 2, 4, 6

[5] OpenAI. Gpt-4, 2023. Accessed: 2024-07-17. 2

[6] 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 *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023. 4

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 4

[8] Yujun Shi, Chuhui Xue, Jun Hao Liew, Jiachun Pan, Hanshu Yan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8839–8849, 2024. 2, 4, 6

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 8

[10] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *Advances in Neural Information Processing Systems*, 36:1363–1389, 2023. 9

[11] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 8

[12] Zewei Zhang, Huan Liu, Jun Chen, and Xiangyu Xu. Gooddrag: Towards good practices for drag editing with diffusion models. *arXiv preprint arXiv:2404.07206*, 2024. 2, 4, 6