# FaceEditor: Text-Driven and Mask-Constrained Face Attribute Editing

Lin Zhang[a,b], Huihuang Zhao[b,c,*], Weiliang Meng[d], Yuan Yang[b], Paul L. Rosin[e], Yu-Kun Lai[e], Yaonan Wang[c]

[a]*School of Intelligence and Computing, Tianjin University, 300072, China*
[b]*School of Computer Science and Technology, Hengyang Normal University, 421002, China*
[c]*National Engineering Laboratory for Robot Visual Perception and Control Technology, Hunan University, China*
[d]*State Key Laboratory of Multimodal Artificial Intelligence Systems at Institute of Automation, Chinese Academy of Sciences, 100190, China*
[e]*School of Computer Science & Informatics, Cardiff University, UK*

**Abstract**

Face image editing has advanced in recent years, with most methods using multimodal conditional guidance to achieve realistic results. However, these methods cannot intuitively edit specific face image regions, and entangled semantics make preserving unrelated attributes difficult. To address these problems, this paper proposes a unified image manipulation framework named FaceEditor, which supports both text and masks for individually or jointly editing facial attributes. The key idea is to train a coarse-to-fine Editing Direction Mapper (ED Mapper) to predict latent manipulation directions from text. During inference, segmentation masks constrain the blending of latent codes and editing features in the feature space, enabling localized and controllable image editing. Additionally, we designed a Global Modulation Module (GMM) to globally blend and optimize latent features at different levels, further enhancing the model's disentangled manipulation capabilities and editing precision. Experimental results show that FaceEditor outperforms existing methods in accuracy, visual realism, and preservation of unrelated attributes, while enabling real-time text-only editing at 0.61s per image. Code is available at https://github.com/Zlin0530/FaceEditor.

*Corresponding author
*Email addresses:* zlin_27@tju.edu.cn (Lin Zhang), happyday.huihuang@gmail.com (Huihuang Zhao), weiliang.meng@ia.ac.cn (Weiliang Meng), yang_yuan2023@163.com (Yuan Yang), RosinPL@cardiff.ac.uk (Paul L. Rosin), LaiY4@cardiff.ac.uk (Yu-Kun Lai), yaonan@hnu.edu.cn (Yaonan Wang)

*Keywords:* Controllable editing, Image generation, Disentangled manipulation, Multimodal, GAN inversion

## 1. Introduction

Text-driven image manipulation, as an important and challenging task in the field of image processing, has long attracted widespread interest and research from scholars. Recently, the great success of cross-modal vision-language joint representation techniques [1, 2, 3] has opened up many possibilities for text-driven image editing operations. Many text-guided image editing methods [4, 5, 6, 7] based on GAN [8] and CLIP [1] can produce satisfactory editing results. These methods predominantly use optimization-based iterative approaches [9, 10, 11] and mapping network learning approaches [7, 12, 13] to accomplish interactive text-based image editing tasks. Additionally, the immense potential of diffusion models in generative capabilities has led many researchers to apply them to fields such as image synthesis and text generation [3, 14, 15, 16, 17], achieving remarkable results. DiffusionCLIP [18] combines diffusion models with CLIP-guided loss for image generation. InstructDiffusion [19] unifies multiple NLP tasks within a single framework, using instructions to perform various visual tasks, such as image editing and enhancement. However, these methods cannot intuitively and flexibly edit specific regions of an image and often encounter editing failures. For example, it is very challenging for these methods to achieve tasks such as turning the upper lip of the input source face image red or lightening the skin on the left half of the face. Additionally, since the visual attributes of different parts of an image are not independent of each other, accurately and effectively completing image editing tasks based on the relevant text descriptions while maintaining attributes unrelated to the text description is extremely challenging.

We believe that solving this issue requires identifying and precisely editing the relevant regions of the source image based on text prompts. Despite significant advancements in image editing, no existing work provides a unified framework that supports both text-only and text + mask local editing for facial attributes. Current methods, such as HairCLIPv2 [11], offer interaction modes like text, sketches, and masks, but

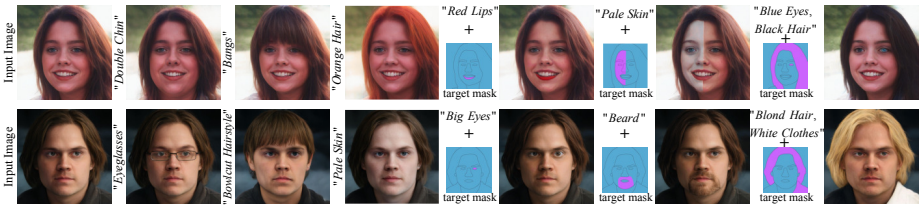


Figure 1: Examples of image manipulation with FaceEditor, where editing conditions can come from text or a combination of text and segmentation masks.

are limited to specific regions, such as hair, and do not extend to other facial features like the eyes, mouth, and skin. Therefore, exploring how to intuitively, succinctly, and precisely manipulate any region of the source face image is an urgent and challenging problem that needs to be addressed.

To achieve this goal, we propose FaceEditor, the first unified face image editing framework that supports both text and mask controls. This framework enables face image manipulation either based solely on text prompts or in conjunction with user-provided masks for specific regions of the face image. Specifically, inspired by DeltaEdit's [13] approach of training without text, we use image pairs as pseudo-texts during training to improve model efficiency and generalization. This is well-suited for face attribute editing, where fine-grained local edits are needed for regions like the eyes, mouth, and skin. Traditional image-text pairs require separate text descriptions for each region, which is labor-intensive. Using image pairs allows the model to learn from feature differences, eliminating the need for extensive text annotations. The key to training is learning a coarse-to-fine mapping network, ED Mapper, based on these differences to predict changes in the latent code $s$. Since the CLIP features of text-image differences imply similar semantic changes, during the inference phase, we use text feature differences through the ED Mapper to predict the editing direction of the source image's latent code $s$, resulting in intermediate editing results that align with the text description. This method improves the model's precision in localized edits, enhances generalization to unseen tasks, reduces reliance on manual annotations, and mitigates overfitting, enabling adaptation to new fine-grained editing tasks. Additionally, to improve the accuracy of intermediate editing results and prevent failures in text-based

Table 1: Comparison of different methods in terms of interaction modes and editing capabilities. Our method uniquely supports both text-only and text + mask interaction modes, enabling both global and local attribute editing.

| | Hair Transfer | Face Editing | Local Hair Transfer | Local Face Editing | Text | Text & Mask |
|---|---|---|---|---|---|---|
| StyleCLIP | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ |
| TediGAN | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ |
| DeltaEdit | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ |
| HairManip | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| HairCLIP | ✓ | ✗ | ✗ | ✗ | ✓ | ✗ |
| HairCLIPv2 | ✓ | ✗ | ✓ | ✗ | ✓ | ✓ |
| DiffusionCLIP | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ |
| InstructDiffusion | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ |
| Ours | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

edits, we designed a Global Modulation Module (GMM) that blends and optimizes features across different semantic levels of latent code information and editing conditions. Many methods have demonstrated that performing feature blending in the $\mathcal{FS}$ embedding space can achieve better reconstruction and preserve local details [11, 20, 21]. Therefore, we map the intermediate editing results to the $\mathcal{FS}$ embedding space based on the optional segmentation mask. Then, we optimize the intermediate editing results according to the specified mask for the regions to be edited, while ensuring that the non-edited areas remain consistent with the source image, thereby generating the final edited result image.

To demonstrate the superiority of FaceEditor, we conducted extensive quantitative and qualitative comparisons as well as user studies. The numerous experimental results indicate that FaceEditor excels in training and inference speed and flexibility, manipulation accuracy, and the visual naturalness and realism of the edited results. Figure 1 illustrates the specific experimental results of face image manipulation using text and masks with FaceEditor, and Table 1 summarizes the interaction modes and editing capabilities supported by different methods.

Overall, our main contributions can be summarized as follows:

- To address the limitations of text-driven editing, which lacks explicit spatial constraints and may unintentionally affect irrelevant regions, as well as mask-based editing, which provides limited semantic expressiveness for complex attribute changes, we propose FaceEditor, a unified framework that integrates text prompts and segmentation masks to achieve semantically driven and spatially controllable facial attribute editing.
- We propose a unified disentanglement strategy that jointly integrates a coarse-to-fine ED Mapper and $\mathcal{FS}$-space blending method. Through progressive latent feature refinement, global modulation, and spatially guided feature disentanglement and blending, the framework achieves precise, controllable, and globally consistent image manipulation.
- We demonstrate a real-time editing capability with results produced in just 0.61 seconds, making our method highly suitable for practical applications where speed is critical.
- We design task-specific loss functions that improve manipulation accuracy and enhance the visual realism of the generated images.

## 2. Related Work

### *2.1. Latent Space Image Manipulation*

Mapping images into the latent space using GAN inversion for understanding and manipulation has become an active research area for image reconstruction and editing tasks [22, 23, 24, 25]. With the great success of StyleGAN's latent space in feature disentanglement and semantic representation of images, many models have been proposed to disentangle and edit latent features. Among them, the e4e [22] inversion framework is widely used for image editing tasks due to its ability to obtain high-quality editable latent codes [5, 13, 7]. However, the latent codes obtained through e4e inversion still suffer from feature entanglement, which means that changing one dimension can affect

multiple attributes during image editing tasks. To address the issue of feature entanglement, Barbershop [20] proposed a novel $\mathcal{FS}$ latent space. This method optimizes the latent code by combining a structure tensor $\mathcal{F}$ and an appearance code $\mathcal{S}$, enabling more precise and detailed editing. Inspired by this, we seek changes in the latent code $s$ within the feature space $S$ and optimize the editing proxy features in the $\mathcal{FS}$ feature space to develop an image editing framework that supports both text and mask conditions.

### *2.2. Text-driven Image Manipulation*

Text-driven image editing [26, 27, 28, 29] uses given text descriptions to edit source images and generate high-fidelity results that align with the text descriptions. CLIP [1] serves as a bridge between text and image representations, and its emergence has led many researchers to explore text-based manipulation [30, 31, 32]. The innovative research StyleCLIP [10] combines the powerful text-image representation capability of the CLIP [1] model with the high-quality image generation model StyleGAN [33] to achieve interactive text-driven image editing tasks. However, editing an image with StyleCLIP requires several minutes of optimization time and the dynamic adjustment of complex hyperparameters to improve the quality of the generated image, which is very time-consuming and inflexible. TediGAN [9] encodes images and text into the latent space for style mixing, generating images that conform to the given text descriptions. FFCLIP [12] performs semantic alignment and injection between text and images in the Semantic Modulation Block to achieve facial attribute manipulation, enhancing the model's generalization and flexibility. HairCLIP [5] uses 44 text prompts and images for training to achieve joint hair editing. DeltaEdit [13] proposes using images as pseudo-text for training to address the costly image-text annotation and the inflexibility during training or inference. However, DeltaEdit cannot edit specified regions of the image and suffers from mismatches between text prompts and the resulting images, significantly limiting the flexibility and accuracy of the image editing process. Unlike existing methods, we propose an innovative image editing framework that supports text-only editing or joint editing with masks to process specified regions of an image. This offers a more flexible and intuitive interaction mode. Additionally, thanks

to the carefully designed network architecture and loss functions, our method demonstrates good performance in both inference speed and editing effectiveness.

## 3. Method

### *3.1. Overview*

Our goal is to achieve precise and efficient face image manipulation. To this end, it is natural for us to abandon the use of large amounts of manually annotated image-text data for training manipulation models from scratch, as this is very time-consuming, inflexible, and does not generalize well to unseen text. Benefiting from the development of the cross-modal CLIP model [1], the representation between language and images has become possible. Studies have shown that the CLIP feature differences of paired visual-text data are semantically similar [10, 12, 13, 18, 34], meaning that changes in text features align with corresponding alterations in image representations. This property allows the model to generalize to unseen text prompts during inference, as the learned features can effectively map new text descriptions to latent image spaces. Additionally, we rely on CLIP's ability to transfer semantic information from previously learned text-image pairs to perform zero-shot inference with novel prompts.

### *3.2. FaceEditor*

Our method aims to precisely and controllably edit the target image based on input text prompts and optional masks. As shown in Figure 2, FaceEditor is divided into two main parts: training and inference. During the training phase, we randomly select two images from the training dataset as the source image $I_1$ and the target image $I_2$. We use the CLIP image encoder and the StyleGAN Inversion [22] method to extract their respective CLIP image embeddings $i_1$ and $i_2$ as well as their latent codes $s_1$ and $s_2$ in the latent space $S$. We then calculate the visual feature difference $\Delta i = i_2 - i_1$ between the image embeddings $i_1$ and $i_2$, and concatenation it with the image embedding $i_1$ to obtain the editing vector $i$. Next, we divide the latent code $s_1 \in \mathbb{R}^{6048}$ and the editing vector $i \in \mathbb{R}^{1024}$ into three levels: coarse$(s_c, i_c)$, medium$(s_m, i_m)$, and fine$(s_f, i_f)$.

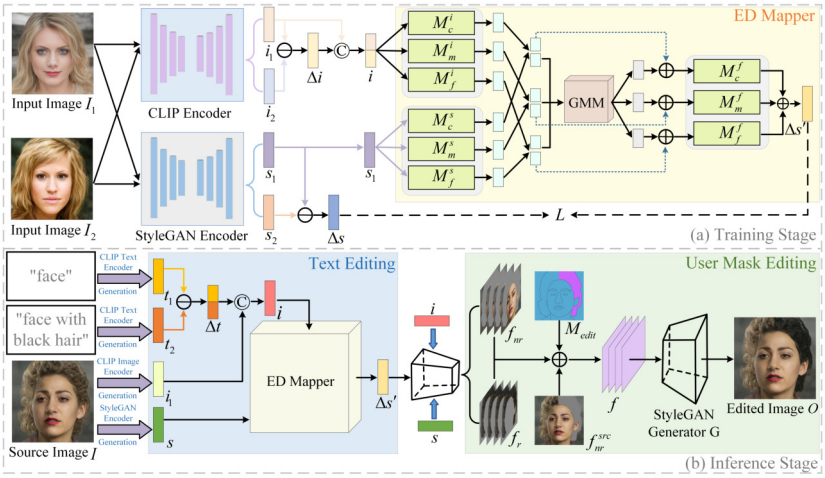


Figure 2: Overview of the FaceEditor framework. FaceEditor is divided into training and inference phases. (a) Training phase: Two randomly selected images are encoded into the CLIP image space and the StyleGAN latent space $S$. The latent embeddings $s_1$ and $i$ are fed into the ED Mapper to learn the predicted manipulation direction $\Delta s'$, constrained by a loss function. (b) Inference phase: FaceEditor uses the trained ED Mapper to calculate the editing direction $\Delta s'$ and manipulates specific regions based on the input optional mask.

Specifically, $s_c \in \mathbb{R}^{3\times512}$ and $s_m \in \mathbb{R}^{4\times512}$ correspond to the coarse- and medium-level style components, respectively, while $s_f \in \mathbb{R}^{2464}$ represents the remaining high-resolution style components. We then feed these into the ED Mapper to predict the manipulation direction $\Delta s'$. Finally, by calculating the difference between $\Delta s'$ and the latent code change $\Delta s = s_2 - s_1$, we continuously adjust and optimize the manipulation direction $\Delta s'$, thereby enhancing the editing accuracy and the quality of the generated images. Mathematically, they follow the following formula:

$$(i_1, i_2) = E_{CI}(I_1, I_2), (s_1, s_2) = E_{SI}(I_1, I_2), \Delta s' = FM(s_1, i_1, \Delta i) \tag{1}$$

where $E_{CI}$, $E_{SI}$ and $FM$ represent the CLIP image encoder, StyleGAN Inversion Encoder, and ED Mapper, respectively.

Due to the semantic similarity between CLIP's textual and image feature differences, during the inference phase, we use the text feature difference $\Delta t = t_2 - t_1$ instead of the image feature difference $\Delta i$ used during the training phase to perform face image editing tasks. More specifically, we use the CLIP image encoder and the StyleGAN

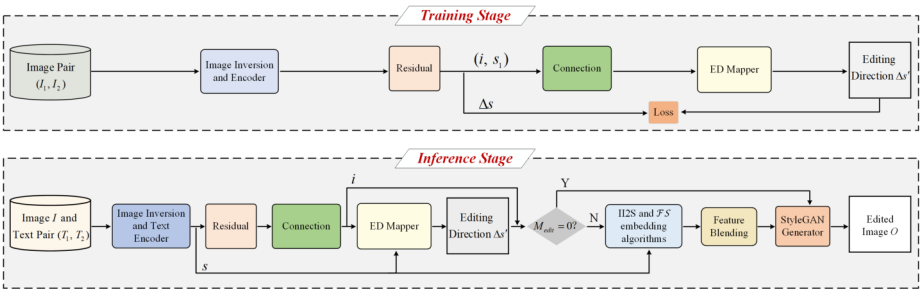


Figure 3: Simplified overview of the FaceEditor training and inference pipeline.

inversion method to obtain the image embedding $i_1$ and the latent code $s$ of the source image $I$. Next, we use the CLIP text encoder to extract the text embeddings $t_2$ and $t_1$ for the target and source texts, respectively, and compute the text feature difference $\Delta t = t_2 - t_1$. Then, we concatenate image embedding $i_1$ with text feature difference $\Delta t$ to obtain $i$, and feed $i$ along with latent code $s$ into the trained ED Mapper to predict the change $\Delta s'$ in latent code $s$. During the inference phase, the predicted changes $\Delta s'$ follow the formula:

$$\Delta s' = FM(s, i_1, \Delta t) \tag{2}$$

where $s$ and $i_1$ represent the latent code $s$ and image embedding $i_1$ obtained from the source image using the StyleGAN Inversion Encoder and CLIP image encoder, respectively. $\Delta t = t_2 - t_1$ denotes the text feature difference between the target text embedding and the source text embedding.

Next, if the user provides a mask for the region to be edited, FaceEditor will use the II2S [35] and $\mathcal{FS}$ embedding algorithms [20] to obtain the latent code features of the intermediate editing result $O_m$ in the $\mathcal{FS}$ embedding space, and separate it into relevant semantic attribute features $f_r$ and unrelated semantic attribute features $f_{nr}$. Then, based on the segmentation mask, we mix $f_r$ and $f_{nr}$ with the unrelated semantic attribute features $f_{nr}^{src}$ of the source image to obtain the final latent code features $f$. Finally, we input the latent code features $f$ into the StyleGAN Generator $G$ to generate the final resultant image $O_f$. Figure 3 presents a simplified schematic of the FaceEditor training and inference pipeline.

**ED Mapper.** The key to training is to learn an ED Mapper that accurately and pre-

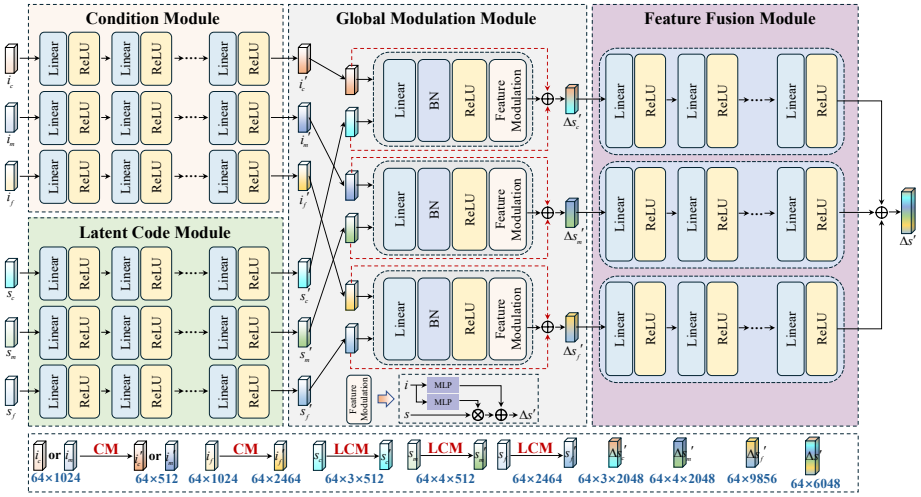


Figure 4: Details of the ED Mapper. The purpose of the ED Mapper is to predict the editing direction based on text prompts. The bottom of the image illustrates the variations across different vector dimensions.

cisely predicts the manipulation direction of the latent code based on text prompts. The specific architecture of the ED Mapper is shown in Figure 4. Our ED Mapper is mainly composed of the Condition Module (CM), Latent Code Module (LCM), Global Modulation Module (GMM), and Feature Fusion Module (FFM). The CM, LCM, and FFM are designed with three levels: coarse, medium, and fine. Each level consists of five simple fully connected layers that handle latent code features and editing conditions at different semantic levels. The purpose of this design is that many methods [9, 5, 36, 37] have proven that the different levels of facial image features in StyleGAN correspond to different semantic levels. By controlling these levels hierarchically, it is possible to achieve manipulations of facial images at different granularities. To prevent editing failures and improve image editing precision, we designed a unique global modulation module that uses a multilayer perceptron (MLP) to modulate the latent code features through conditional embeddings, enabling controlled image feature editing through condition vectors. The role of the Feature Fusion Module is to integrate the generated coarse-to-fine features and predict the editing direction $\Delta s'$.

We used three types of losses to train the ED Mapper: L2 distance reconstruction

loss $\mathcal{L}_{rec}$, cosine similarity loss $\mathcal{L}_{cos}$, and smooth L1 loss $\mathcal{L}_{sl}$. Specifically, to improve the precision and efficiency of image manipulation, we employ $\mathcal{L}_{rec}$, $\mathcal{L}_{cos}$, and $\mathcal{L}_{sl}$ to effectively constrain and supervise the predicted embedding direction $\Delta s'$, as detailed below:

$$\begin{aligned} \mathcal{L}_{FM} &= \mathcal{L}_{\text{rec}} + \mathcal{L}_{sl} + \mathcal{L}_{\cos} \\ &= \left\|\Delta \text{s}' - \Delta \text{s}\right\|_2 + 1 - \cos(\Delta s', \Delta s) + \mathcal{L}_{\text{sl}} + smooth_{L1}(\Delta s', \Delta s) \end{aligned} \tag{3}$$

where $\mathcal{L}_{FM}$ represents the overall objective loss function used to train the ED Mapper.

**Text Editing.** As shown in Figure 2(b), text editing aims to manipulate the source image based on the input text prompts, thereby generating high-fidelity images that match the text descriptions. Therefore, during the inference process, we convert the input text prompts into text embeddings $t_1$ and $t_2$ in the CLIP vector space, extract the image embedding $i_1$ and latent code $s$ of the source image, and then use the trained ED Mapper to predict the editing direction, thus performing the editing task on the source image. It is worth noting that when using text prompt formats like “face” and “face with xxx", the generated resultant images better conform to the target text prompts, as thoroughly demonstrated in the DeltaEdit [13] paper. Additionally, if only text editing is used for the image, i.e., the input mask $M = 0$, the final output result $O_f$ equals the intermediate edited result $O_m$:

$$O_f = O_m = G(s + \Delta s') \tag{4}$$

where $G$ represents the StyleGAN generator, and $\Delta s'$ is the change in latent code $S$ predicted by the ED Mapper.

In summary, the text-driven editing branch formulates attribute manipulation as latent direction prediction in the S space. By leveraging the semantic difference between source and target text embeddings, the ED Mapper predicts an editing direction that aligns the latent code with target semantics while preserving global identity structure, providing the basis for subsequent mask-constrained refinement.

**Mask Editing.** To achieve precise and controllable image manipulation, we effectively limit the editable regions of the source image by controlling the input mask. Furthermore, we first use the II2S and $\mathcal{FS}$ embedding algorithms to extract the relevant

semantic attribute features $f_r$ and the unrelated semantic attribute features $f_{nr}$ from the intermediate editing result $O_m$, and then mix them with the unrelated semantic attribute features $f_r^{src}$ of the source image $I$. During the feature mixing process, we aim to retain the attribute features of the intermediate editing result in the mask-based editing region while ensuring that the non-editing regions remain consistent with the source image. Therefore, we specifically designed a mask loss to constrain the editing region as follows:

$$\mathcal{L}_{r\_mask} = \left\| \left( O_m * M_{edit} - O_f * M_{edit} \right) \right\|_2^2 \tag{5}$$

where $O_m$ and $O_f$ represent the intermediate edited image generated by text prompts and the final result image, respectively. $M_{edit} = P_i(I)$ denotes the mask of the region to be edited in the source image, obtained from the facial parsing network $P_i$ [38]. Similarly, we add a non-edit region preservation loss $L_{i\_mask}$ to effectively supervise the non-edit regions, ensuring that these regions remain unchanged:

$$\mathcal{L}_{i\_mask} = \|(I * (1 - M_{edit}) - O_f * (1 - M_{edit})\|_2^2 \tag{6}$$

Additionally, we introduce a perceptual loss $\mathcal{L}_{\text{LPIPS}}$ to guide the non-edit regions of the result image to align closely with the source image. The mathematical derivation formula for $\mathcal{L}_{\text{LPIPS}}$ is shown in the appendix.

Therefore, the complete objective loss $\mathcal{L}_{ME}$ for optimizing the features of the intermediate editing results with the combined mask is as follows:

$$\mathcal{L}_{ME} = \lambda_{r\_mask}\mathcal{L}_{r\_mask} + \lambda_{i\_mask}\mathcal{L}_{i\_mask} + \lambda_{\text{LPIPS}}\mathcal{L}_{\text{LPIPS}} \tag{7}$$

where $\lambda_{r\_mask}$, $\lambda_{i\_mask}$, and $\lambda_{\text{LPIPS}}$ are set to 500, 10, 1 respectively by default. These loss weights were empirically tuned through preliminary experiments and are kept fixed for all our experiments.

In summary, the mask-constrained editing branch introduces spatial control by decoupling and fusing relevant and non-relevant features in the $\mathcal{FS}$ embedding space. Guided by the mask, it selectively applies semantic modifications while preserving unaffected regions, thereby improving local precision and boundary consistency.

## 4. Experiments

### *4.1. Implementation Details*

To validate the effectiveness and superiority of the proposed method, we trained and evaluated our model using the FFHQ [33] dataset. For dataset partitioning, we randomly divided the FFHQ dataset into 58,000 images for the training set and 12,000 images for the test set. All training and evaluation were conducted on a single RTX3090 GPU. The model is trained for 100 epochs in total, corresponding to 550,000 iterations. We conduct extensive quantitative and qualitative analyses, user studies, and ablation experiments to validate the effectiveness and generalization ability of the proposed method. More detailed implementation settings are provided in appendix.

### *4.2. Quantitative and Qualitative Comparison*

**Visualization of Image Manipulation Results.** Figure 5 shows the images generated by FaceEditor under different text prompts. Additionally, we use text prompts together with user-provided masks to edit specific image regions. Figure 6 presents the corresponding manipulation results, where the second and fifth columns indicate the editing conditions. The segmentation masks are extracted from the source images using a facial parsing network [38], with the purple regions denoting the selected editing areas. As shown in Figures 5 and 6, FaceEditor performs realistic local edits guided by text prompts while preserving unrelated attributes and non-edited regions of the image. Additional visual results are provided in the appendix.

**Comparison with Text-driven Image Manipulation Methods.** To validate the effectiveness of the proposed method in text-guided image processing, we compare FaceEditor with the latest text-guided image processing methods, including StyleCLIP [10], TediGAN [9], DeltaEdit [13], DiffusionCLIP [18], and InstructDiffusion [19]. Figure 7 presents the specific comparison results generated by six different text descriptions. During the editing process, we follow the evaluation settings of StyleCLIP and TediGAN. The disentanglement threshold parameter of StyleCLIP is set to the default value of 0.15, and the manipulation strength parameter $\alpha$ is gradually increased from 2 to 4 in increments of 1. For TediGAN, the $loss\_clip\_weight$ is set to 1.0, and the optimization iteration count is set to the default value of 200.

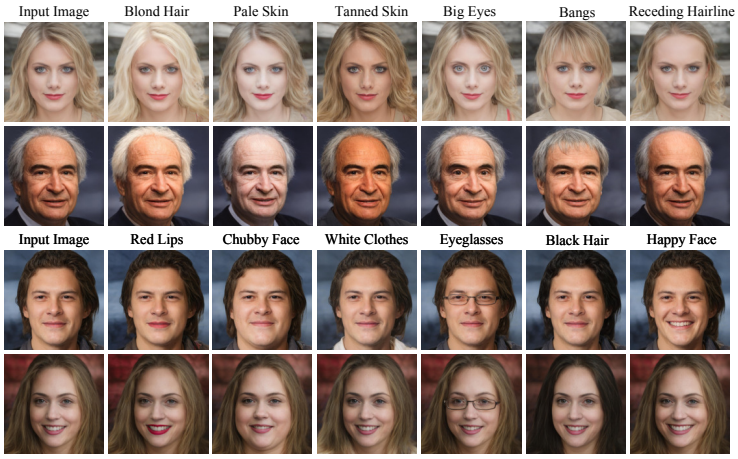


Figure 5: Various face image editing results generated by FaceEditor using only text prompts, with key descriptions from the text prompts listed above each group of images.

Table 2: Quantitative comparison results with current state-of-the-art methods, where higher PSNR, SSIM, and IDS scores indicate better performance.

| Methods | PSNR↑ | SSIM↑ | IDS↑ |
|---|---|---|---|
| StyleCLIP $\alpha = 2$ | 24.76±1.48 | 0.73±0.05 | 0.87±0.03 |
| StyleCLIP $\alpha = 3$ | 24.00±2.25 | 0.72±0.05 | 0.84±0.07 |
| StyleCLIP $\alpha = 4$ | 22.88±3.02 | 0.70±0.04 | 0.80±0.09 |
| TediGAN | 10.26±1.86 | 0.33±0.08 | 0.42±0.11 |
| DeltaEdit | 23.38±4.59 | 0.83±0.07 | 0.87±0.07 |
| DiffusionCLIP | 9.90±1.45 | 0.23±0.05 | 0.54±0.12 |
| InstructDiffusion | 22.37±7.06 | 0.77±0.07 | 0.79±0.18 |
| **Ours** | **25.91**±2.18 | **0.88**±0.08 | **0.88**±0.02 |

As shown in Figure 7, the results generated by StyleCLIP are unstable under different manipulation strengths $\alpha$, and it is difficult to balance diverse text editing tasks with a single set of hyperparameters. Due to limited disentanglement, TediGAN and DiffusionCLIP often fail to accurately follow the intended manipulation directions, leading to undesired changes in unrelated attributes such as identity and background. DeltaEdit improves editing flexibility by learning the mapping between image and text feature variations through a Delta Mapper; however, projecting these variations into


Figure 6: FaceEditor manipulates images using both text and masks. The text and mask editing conditions are displayed to the right of each input image. The top shows a zoomed-in view of the key region.

the style space may lead to information loss or incomplete feature capture. The results produced by InstructDiffusion are also unstable and sensitive to instruction variations. In contrast, FaceEditor consistently produces more stable and realistic editing results across different textual descriptions, while better preserving unrelated attributes.

To objectively and rigorously evaluate the quality of the images generated by the proposed method, we report the quantitative results of PSNR, SSIM, and IDS under ten different text prompt conditions. For each text prompt, 200 edited images are generated, resulting in a total of 2,000 samples for evaluation. All quantitative metrics are computed on the inverted and spatially aligned images to ensure fair comparison across


Figure 7: Visual comparison between FaceEditor and existing methods for image manipulation using text descriptions. The target attributes from the text prompts are shown on the far left. The upper-right corner presents a zoomed-in view of the key region.

different methods. It is worth noting that, to further explore the impact of StyleCLIP's manipulation strength parameter $\alpha$ on the generated results, we set the parameter $\alpha$ to 2, 3, and 4, respectively, and compared these settings with the proposed method to verify the superiority of FaceEditor. The specific experimental results are shown in Table 2. All comparison results are reported as the overall mean and standard deviation computed over ten edited text prompts. As shown in Table 2, our method outperforms the state-of-the-art comparison methods across all metrics.

**Comparison with Hair Manipulation Methods.** The complexity and diversity of hair texture and geometry make hair editing a challenging task. To further validate the robustness and superiority of the proposed method, we compare FaceEditor with the state-of-the-art text-driven hair transfer methods, including HairManip [7], HairCLIP [5], and HairCLIPv2 [11]. The specific comparison results are shown in Figure 8. As can be seen from Figure 8, HairManip performs well in most hair editing tasks,

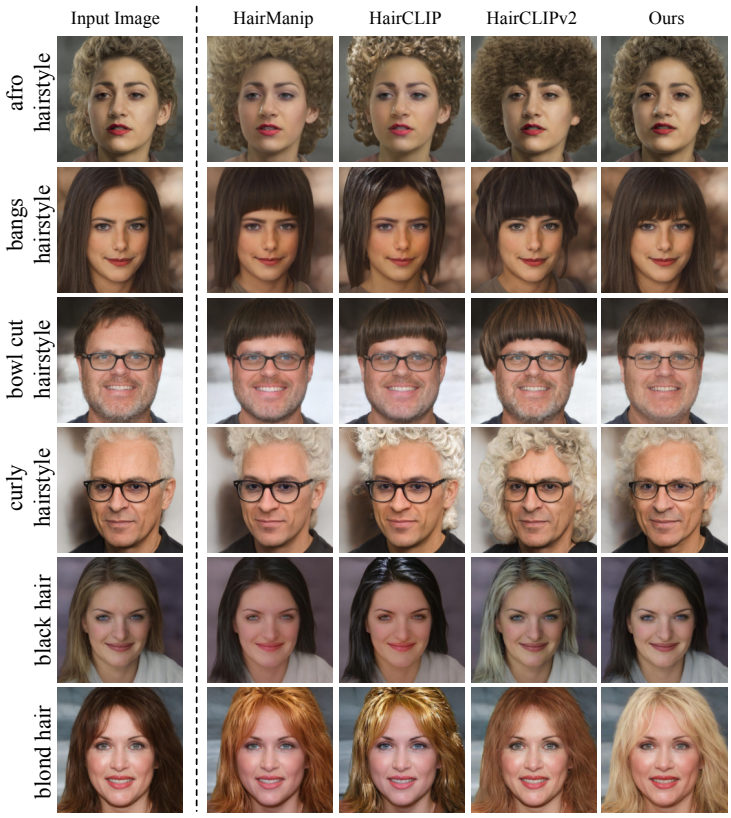


Figure 8: Qualitative comparison of FaceEditor with the current state-of-the-art hair editing methods. The text-based hair editing prompts are displayed on the far left.

but it easily modifies unrelated attributes in the generated results, such as the face and background. HairCLIP produces results with white spots in most cases, which is consistent with the conclusions drawn in the HairManip paper. Although HairCLIPv2 can effectively preserve unrelated attribute features, it struggles with handling complex boundary areas between facial components and hair, resulting in generated images that lack naturalness (see the fourth row, “curly hairstyle"), and sometimes even produce failed results. It is worth mentioning that, compared to the latest methods specifically designed for hair editing tasks, our approach produces the most satisfactory and impressive results in terms of manipulation accuracy, preservation of unrelated attributes, and visual naturalness.

Table 3: User study on manipulation accuracy, preservation of unrelated attributes, and visual naturalness for the three comparison types. Fleiss' $\kappa$ is reported to measure inter-rater agreement among the 20 participants.

| | Metrics | | |
|---|---|---|---|
| Fleiss' $\kappa$ / Methods | Accuracy | Preservation | Naturalness |
| **Fleiss' $\kappa$ (Text-Driven)** | 0.64 | 0.63 | 0.58 |
| StyleCLIP $\alpha = 2$ | 6.3% | 6.0% | 10.6% |
| StyleCLIP $\alpha = 3$ | 8.5% | 8.2% | 5.9% |
| StyleCLIP $\alpha = 4$ | 19.6% | 7.3% | 7.8% |
| TediGAN | 2.8% | 0.4% | 1.7% |
| DeltaEdit | 23.7% | 11.9% | 19.9% |
| **Ours** | **39.1%** | **66.2%** | **54.1%** |
| **Fleiss' $\kappa$ (Hair Manipulation)** | 0.46 | 0.57 | 0.45 |
| HairManip | 29.7% | 7.6% | 24.1% |
| HairCLIP | 11.2% | 6.2% | 16.8% |
| HairCLIPv2 | 24.5% | 41.9% | 21.3% |
| **Ours** | **34.6%** | **44.3%** | **37.8%** |
| **Fleiss' $\kappa$ (Diffusion Model)** | 0.42 | 0.69 | 0.55 |
| DiffusionCLIP | 4.4% | 1.9% | 6.3% |
| InstructDiffusion | 28.1% | 20.3% | 20.1% |
| **Ours** | **67.5%** | **77.8%** | **73.6%** |

### *4.3. User Study*

In the user study, we invited 20 volunteers with research backgrounds in computer vision to conduct a subjective evaluation of the generated results for the three comparison types: Text-Driven, Hair Manipulation, and Diffusion Model. For the above three comparison methods, we randomly selected 40 test samples from ten text descriptions excluding hair edits and 20 samples from ten text descriptions specifically for hair edits, resulting in a total of 100 test samples. Before the evaluation, the results produced by different methods in each test sample were anonymized and presented in a randomly shuffled order. For each test sample, the results generated by all competing methods were displayed simultaneously, and participants were asked to independently select the best result according to three evaluation criteria: editing accuracy, preservation of unrelated attributes, and visual naturalness.

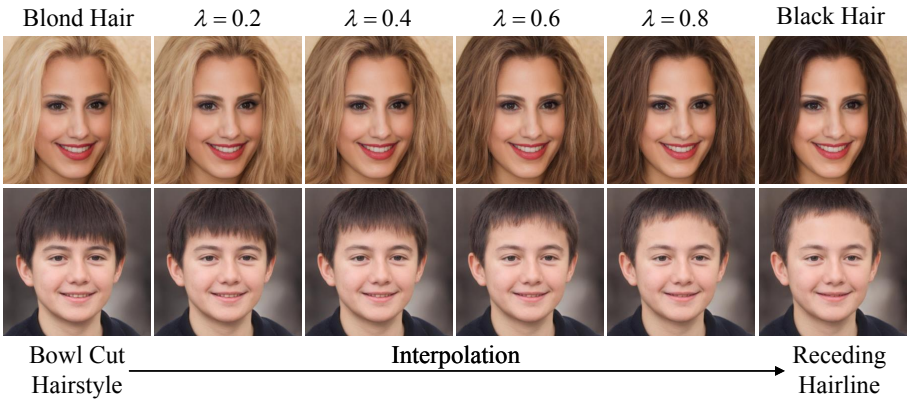


Figure 9: Results of the attribute interpolation experiment. By controlling the blending weight $\lambda$ from 0 to 1, the interpolated images in the first row transition smoothly from the attribute "Blond Hair" to "Black Hair", while the second row transitions smoothly from "Bowl Cut Hairstyle" to "Receding Hairline".

To further assess the reliability of the subjective evaluation, we report the inter-rater agreement using Fleiss' $\kappa$, which measures the consistency among multiple raters. As shown in Table 3, the obtained $\kappa$ values range from 0.42 to 0.69 across different comparison settings, indicating moderate to substantial agreement among the evaluators. The results also show that our method consistently achieves the highest preference scores in terms of manipulation accuracy, preservation of unrelated attributes, and visual naturalness compared with the competing methods.

### *4.4. Attributes Interpolation*

To verify that FaceEditor can perform fine-grained attribute editing on facial images, we randomly selected two edited latent codes, $s_a, s_b \in S$, to conduct an attribute interpolation experiment. Specifically, we blend latent codes $s_a$ and $s_b$ through linear weighting to obtain an intermediate latent code $s_o = \lambda s_a + (1 - \lambda) s_b$. As shown in Figure 9, we gradually increase the mixing parameter $\lambda$ from 0 to 1 (in intervals of 0.2), resulting in the interpolated image's semantic attributes transitioning from the initial attribute "Blond Hair" ("Bowl Cut Hairstyle") to "Black Hair" ("Receding Hairline").

### *4.5. Efficiency Analysis*

To further evaluate the efficiency and flexibility of the proposed method, we tested the time consumption of FaceEditor and six comparison methods. It is worth noting that, due to the imbalance in computational resource consumption between diffusion models and the other two comparison methods, we do not conduct Efficiency Analysis testing for the diffusion models here. Table 4 provides a comprehensive comparison of different methods in terms of preprocessing time, training time, inference time, inference memory usage (batch size = 1), supported image resolution (Res.), and latent space. Furthermore, we separately evaluate the efficiency of our method under two settings: text-only manipulation (Ours(T)) and joint text-and-mask editing (Ours(T+M)).

As shown in Table 4, although StyleCLIP (Global Directions) [10] does not require training, it takes 4 hours to predict global manipulation directions. Additionally, during the inference phase, it still requires approximately 10 seconds to manually adjust hyperparameters to achieve satisfactory manipulation results. The extended inference times of TediGAN [9] and HairCLIPv2 makes them difficult to achieve real-time image editing tasks. HairManip and HairCLIP both take more than 30 hours to train the model. Different from them, the proposed method can generate manipulation results that conform to text prompts in real time in about 0.61s once training is completed. Moreover, the proposed text and mask joint editing method, Ours(T+M), requires only about 15 seconds (with 8 seconds spent manually selecting the editing mask) to edit any region of a facial image, a capability not provided by any of the other methods. It is worth noting that the training and inference efficiency of the proposed method generally surpasses that of baseline methods, except for the DeltaEdit [13] method. However, our goal is not to achieve the utmost computational efficiency but to ensure high precision and controllable image manipulation. Therefore, the results of efficiency analysis, while achieving accurate and controllable editing tasks, are acceptable.

### *4.6. Ablation Analysis*

**Importance of Various Losses.** To verify the effectiveness and necessity of each loss function for the model, we alternately remove the L2 distance reconstruction loss

Table 4: Time efficiency analysis of the proposed method compared to the latest methods. $T_1$ represents the time cost associated with manually adjusting StyleCLIP hyperparameters, which is approximately 10 seconds per case. $T_2$ represents the time cost for manually determining the mask for the editing region, which is approximately 8 seconds per case.

| | Pre-proc. | Training time | Infer. time | Infer. memory | Res. | Latent space |
|---|---|---|---|---|---|---|
| StyleCLIP | 4h | - | 8.09s+$T_1$ | 5.1GB | 1024 | $\mathcal{S}$ |
| DeltaEdit | 4h | 2h+ | 0.47s | 13.0GB | 1024 | $\mathcal{S}$ |
| TediGAN | - | 12h+ | 22.32s | 3.6GB | 256 | $\mathcal{W}$+ |
| HairManip | 2h+ | 32h+ | 3.80s | 3.7GB | 1024 | $\mathcal{W}$+ |
| HairCLIP | 2h+ | 31h+ | 3.70s | 3.7GB | 1024 | $\mathcal{W}$+ |
| HairCLIPv2 | 2h | - | 25.28s | 14.8GB | 1024 | $\mathcal{FS}$ |
| Ours(T) | 6h | 10h | 0.61s | 12.8GB | 1024 | $\mathcal{S}$ |
| Ours(T+M) | 7h+ | 10h | 7.73s+$T_2$ | 16.9GB | 1024 | $\mathcal{S}/\mathcal{FS}$ |

$\mathcal{L}_{rec}$, cosine similarity loss $\mathcal{L}_{cos}$, and smooth L1 loss $\mathcal{L}_{sl}$, while keeping all other components unchanged. Figure 10 shows the specific ablation comparison results. As seen in Figure 10, the smooth L1 loss $\mathcal{L}_{sl}$ helps improve editing accuracy, while the reconstruction loss $\mathcal{L}_{rec}$ and similarity loss $\mathcal{L}_{cos}$ contribute to enhancing the quality of the generated images, particularly in preserving unrelated attributes like identity information. This is because these two losses effectively constrain the editing direction, ensuring that the differences between image and text features are well-aligned in the CLIP space. As a result, the image can be manipulated according to the editing conditions while preserving the stability of unrelated features.

## 5. Conclusion

In this paper, we propose a unified image editing framework, FaceEditor, which supports local and controllable editing of any region of an image using both text descriptions and segmentation masks. The method first predicts the editing direction of the text embedding using a trained ED Mapper. Then, FaceEditor selectively blends and optimizes latent code features with the editing direction in the $\mathcal{FS}$ embedding space, guided by the effective information from the mask, to achieve localized image

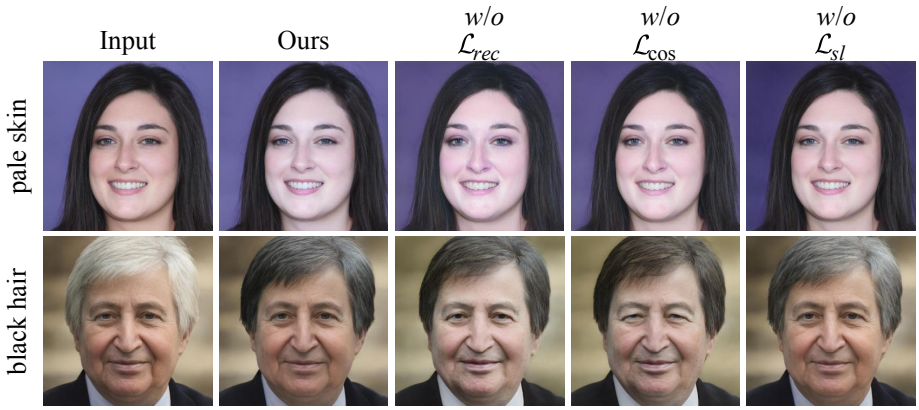


Figure 10: Ablation study comparison of various losses.

manipulation tasks. Additionally, our GMM performs global fusion and modulation of latent code features and editing conditions at different levels, improving the precision and efficiency of the model's manipulation capabilities. Compared to mainstream image manipulation methods, FaceEditor can accurately and realistically edit attributes using only text descriptions, and it can also perform effective and detailed local manipulations based on the selected regions with a mask, which has never been achieved before. Extensive experiments demonstrate that our innovative method offers significant advantages in terms of editing effectiveness and controllability.

**Declaration of Competing Interest** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Acknowledgments** This work was supported in part by the National Natural Science Foundation of China under Grant 61772179 and Grant 12442056, in part by Hunan Provincial Natural Science Foundation of China under Grant 2024JJ5059 and Grant 2023JJ50095, in part by Scientific Research Fund of Hunan Provincial Education Department 22B0728.

## Appendix A. Significance Analysis and Additional Editing Results

We conduct paired t-tests between our method and each competing approach on PSNR, SSIM, and IDS over the same set of test samples. It is worth noting that, while quantitative metrics such as PSNR, SSIM, and IDS (identity similarity between the original and edited images measured using ArcFace) are computed over 2,000 samples to ensure stable and reliable performance evaluation, the statistical significance analysis is conducted on a randomly selected subset of 50 paired samples drawn from the same evaluation set for pairwise comparison. This sample size is generally sufficient for paired t-tests while reducing computational overhead. The results are summarized in A.5, where the t-statistics, the corresponding two-sided p-values, and the effect sizes measured by Cohen's $d$ are reported. While the p-values indicate whether the performance differences are statistically significant, Cohen's $d$ further quantifies the magnitude of improvement between methods. Overall, the statistical tests confirm that the performance gains of FaceEditor on PSNR and SSIM are statistically meaningful across most baselines, with medium-to-large effect sizes in many cases, while IDS improvements are more method-dependent. These results suggest that the improvements reported in the main paper are unlikely to arise from random variation and instead reflect consistent performance advantages of the proposed method.

In Figures A.11, A.12, A.13, and A.14, we provide additional visualization results for the FaceEditor method.

## Appendix B. Method Details

### *Appendix B.1. Training and Implementation Details*

This subsection provides additional implementation details for training FaceEditor. To improve the diversity of training samples and better capture semantic variations in the FFHQ dataset, we augment the training data by sampling 300,000 randomly generated images in the $Z$ space using a pre-trained StyleGAN model and performing inversion with the e4e encoder. For each source sample, a pseudo-text pair is constructed by uniformly sampling a second latent code from the training set without attribute

Table A.5: Paired t-test statistical significance analysis between FaceEditor and each baseline method on PSNR, SSIM, and IDS over 50 paired samples. The table reports the t-statistics, corresponding two-sided p-values, and Cohen's *d* effect sizes to quantify the magnitude of improvement.

| Baseline (vs Ours) | PSNR | | | SSIM | | | IDS | | |
|---|---|---|---|---|---|---|---|---|---|
| | *t* | *p* | *d* | *t* | *p* | *d* | *t* | *p* | *d* |
| StyleCLIP | 5.37 | 6.65e-4 | 0.76 | 10.51 | 5.82e-6 | 1.49 | 4.26 | 2.75e-3 | 0.60 |
| TediGAN | 13.41 | 9.13e-7 | 1.90 | 24.64 | 7.84e-9 | 3.48 | 12.03 | 2.09e-6 | 1.70 |
| DeltaEdit | 2.52 | 2.39e-1 | 0.36 | 18.80 | 3.38e-2 | 2.66 | 0.86 | 5.45e-1 | 0.12 |
| DiffusionCLIP | 10.11 | 5.38e-4 | 1.43 | 15.80 | 9.35e-5 | 2.23 | 3.70 | 2.07e-2 | 0.52 |
| InstructDiffusion | 7.91 | 6.13e-4 | 1.12 | 15.01 | 8.34e-5 | 2.12 | 1.74 | 1.56e-1 | 0.25 |


Figure A.11: Visualization of multi-attribute sequential editing results using FaceEditor. The text descriptions are shown on the far left, and the input source image is displayed at the bottom-right corner of the resulting images.

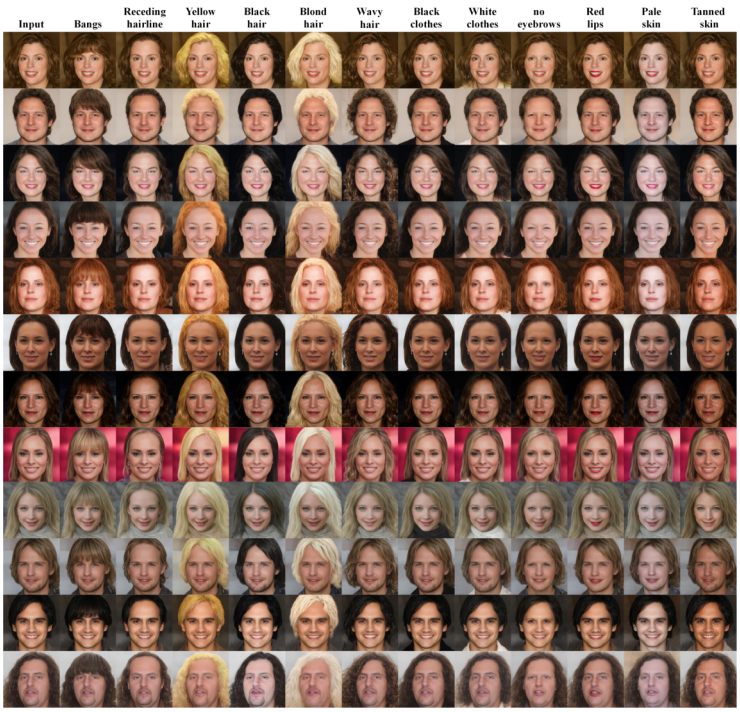


Figure A.12: FaceEditor uses text descriptions to drive image editing. The target attributes in the text descriptions are displayed above each column.

filtering, allowing the resulting latent differences to naturally reflect the semantic distribution of the dataset. A fixed random seed is used for all stochastic operations to ensure reproducibility.

During training, FaceEditor is optimized using the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 0.5 and the batch size is 64. Under this configuration, the training time for each epoch is approximately 0.37 hours, resulting in a total training time of about 37.53 hours for 100 epochs.

*Appendix B.2. Network Architecture*

Table A.6 presents the detailed network architecture of the ED Mapper. The input to the Condition Module (CM) is the editing vector obtained by concatenating the image features with the image/text feature difference. The latent code has a dimensionality of 6048 and is partitioned into different semantic levels, which are then fed into


Figure A.13: FaceEditor uses text descriptions to drive image editing. The target attributes in the text descriptions are displayed above each column.

the Latent Code Module (LCM). Subsequently, together with the editing vector, these features are processed by the Global Modulation Module (GMM) and the Feature Fusion Module (FFM) to produce three level-specific variation vectors. These vectors are finally concatenated to obtain the final manipulation direction with 6048 channels. Additionally, the ED Mapper is implemented using StyleGAN2-based EqualLinear layers with $lr_mul = 0.01$ and Fused LeakyReLU activation. No additional manual weight initialization is applied in the training script, and the parameters follow the default initialization scheme of the corresponding StyleGAN2 layers. For normalization, PixelNorm is applied at the input of each mapping branch, while Batch Normalization is employed within the Global Modulation Module (GMM) to stabilize feature distributions during training.

### *Appendix B.3. Inference Algorithm of FaceEditor*

Algorithm 1 presents the inference pipeline of the proposed FaceEditor framework. During inference, the source image is encoded into the CLIP embedding space and the StyleGAN latent space to obtain $i_1$ and $s$. The source and target texts are encoded to compute the semantic difference $\Delta t = t_2 - t_1$, which is fed into the trained ED Mapper

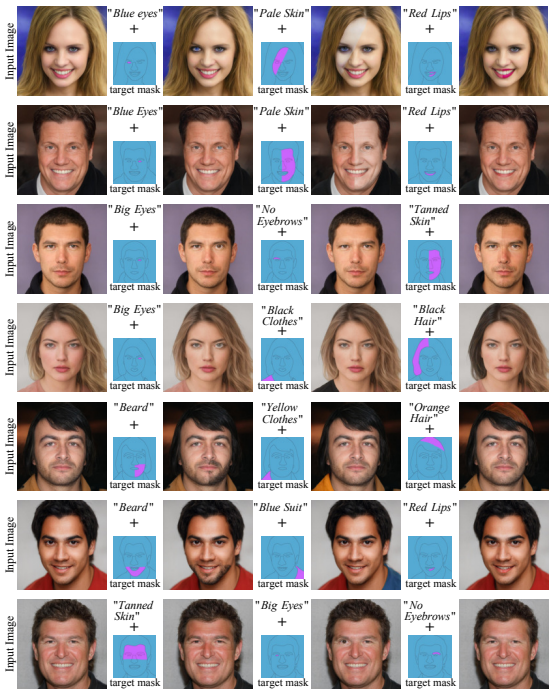


Figure A.14: FaceEditor uses text descriptions and segmentation masks to jointly manipulate images. The editing conditions are displayed between the input and result images.

to predict the editing direction $\Delta s' = F_M(s, i_1, \Delta t)$. The intermediate result is then generated as $O_m = G(s + \Delta s')$.

For text-only editing, $O_m$ is directly taken as the final output. For text+mask editing, the intermediate result is projected into the $\mathcal{FS}$ space, where relevant and unrelated features are separated and fused under mask guidance to preserve non-edited regions. The blended features are finally decoded to produce the edited image $O_f$.

### *Appendix B.4. Perceptual Loss Formulation*

To encourage perceptual consistency in the non-edited regions, we adopt a masked LPIPS-based perceptual loss. The loss is computed between the source image $I$ and the final edited result $O_f$, restricted to the complement of the editing mask $M_{edit}$. Mathematically, it is defined as follows:

$$\mathcal{L}_{\text{LPIPS}} = \sum_{\ell} \frac{1}{H_\ell W_\ell} \sum_{i,j} \| w_\ell^{\text{lpips}} \odot \Delta_{i,j}^{\ell}(I * (1 - M_{edit}), O_f * (1 - M_{edit})) \|^2 \tag{B.1}$$

Table A.6: Detailed architecture of the ED Mapper. "BN" denotes Batch Normalization and "FM" denotes the Feature Modulation module.

| Module | Sub-module | Layers in the module | Input Dim | Output Dim |
|---|---|---|---|---|
| Condition Module | $M_c^i$ | 5×(Linear, ReLU) | (64, 1024) | (64, 512) |
| | $M_m^i$ | 5×(Linear, ReLU) | (64, 1024) | (64, 512) |
| | $M_f^i$ | 5×(Linear, ReLU) | (64, 1024) | (64, 2464) |
| Latent Code Module | $M_c^s$ | 5×(Linear, ReLU) | (64, 3, 512) | (64, 3, 512) |
| | $M_m^s$ | 5×(Linear, ReLU) | (64, 4, 512) | (64, 4, 512) |
| | $M_f^s$ | 5×(Linear, ReLU) | (64, 2464) | (64, 2464) |
| Global Modulation Module | $M_c^g$ | Linear, BN, ReLU, FM | (64, 3, 1024) | (64, 3, 1024) |
| | $M_m^g$ | Linear, BN, ReLU, FM | (64, 4, 1024) | (64, 4, 1024) |
| | $M_f^g$ | Linear, BN, ReLU, FM | (64, 4928) | (64, 4928) |
| Feature Fusion Module | $M_c^f$ | 5×(Linear, ReLU) | (64, 3, 2048) | (64, 3, 512) |
| | $M_m^f$ | 5×(Linear, ReLU) | (64, 4, 2048) | (64, 4, 512) |
| | $M_f^f$ | 5×(Linear, ReLU) | (64, 9856) | (64, 2464) |

**Algorithm 1** Inference Algorithm of FaceEditor.

**Required:** source image $I$; source text $T_1$; target text $T_2$; optional mask $M_{edit}$; CLIP image encoder $E_{CI}$; CLIP text encoder $E_{CT}$; StyleGAN inversion encoder $E_{SI}$; trained ED Mapper $F_M$; StyleGAN generator $G$; $\mathcal{FS}$ embedding algorithms $E_{\mathcal{FS}}$.

$i_1 = E_{CI}(I)$, $s = E_{SI}(I)$;
$t_1 = E_{CT}(T_1)$, $t_2 = E_{CT}(T_2)$;
$\Delta t = t_2 - t_1$;
$\Delta s' = F_M(s, i_1, \Delta t)$;
$O_m = G(s + \Delta s')$;
**if** $M_{edit} = 0$ **then**
    $O_f = O_m$;
    **return:** $O_f$;
**else**
    $(f_{nr}, f_r) = E_{\mathcal{FS}}(s, i_1, \Delta s')$;
    $f = f_r \odot M_{edit} + f_{nr}^{src} \odot (1 - M_{edit})$;
    $O_f = G(f)$;
    **return:** $O_f$;

**Output:** final edited image $O_f$.

where

$$\begin{aligned} &\Delta_{i,j}^{\ell}(I * (1 - M_{edit}), O_f * (1 - M_{edit})) = \\ &\widehat{\mathrm{VGG}}_{i,j}^{\ell}(I * (1 - M_{edit})) - \widehat{\mathrm{VGG}}_{i,j}^{\ell}(O_f * (1 - M_{edit}))) \end{aligned} \tag{B.2}$$

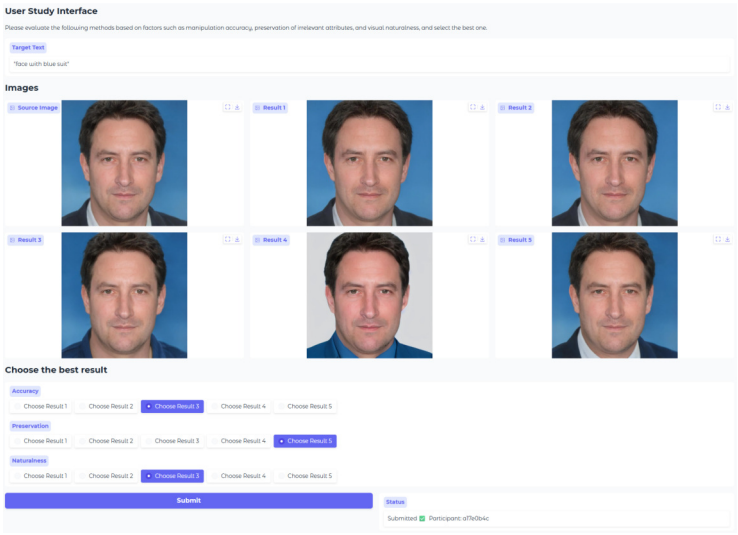


Figure B.15: A visualization example from the user study. Participants were asked to select the best result in terms of manipulation accuracy, attribute preservation, and visual naturalness, given the target prompt and the source image as references.

and the vector $w_\ell^{\text{lpips}}$ represents the learned vector of channel weights of each layer, which is associated with layer $\ell$. $\widehat{\text{VGG}}^\ell$ denotes the activation of layer $\ell$ of the VGG network normalized on the channel dimension.

## Appendix C. User Study Details

In the user study, the presentation order of the five editing methods was randomly shuffled for each sample to eliminate positional bias. All evaluators were blinded to method identities, and no information regarding model names or implementation details was displayed in the interface, thereby preventing subjective preference toward any specific approach, as illustrated in Figure B.15.

## Appendix D. Limitations and Future Work

Despite achieving strong performance in controlled editing scenarios, several limitations remain. First, the method may exhibit variability in detail consistency and visual naturalness in more complex or dynamic editing cases. Second, although FaceEditor

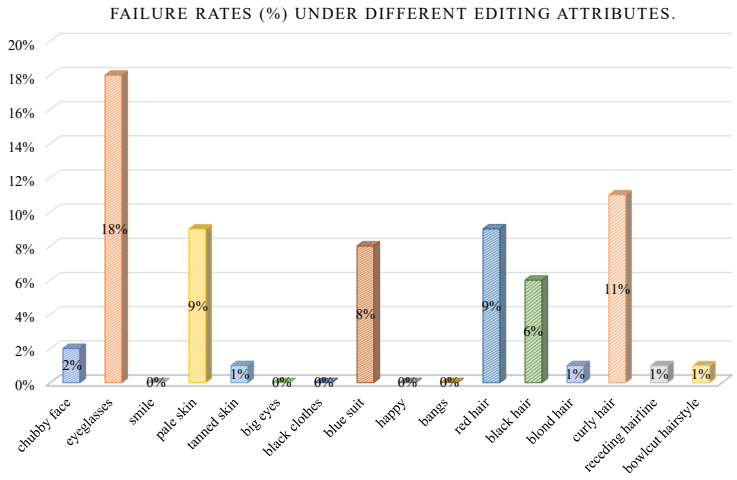


Figure C.16: Failure rates (%) under different editing attributes.

performs well on portrait-centric datasets such as FFHQ, its performance may slightly degrade on non-portrait domains, where generalization becomes more challenging. Third, the editing quality is sensitive to segmentation mask accuracy; lower-quality masks may reduce spatial precision in the edited regions. Additionally, the framework relies on accurate StyleGAN inversion to obtain high-quality latent representations, and errors introduced during the inversion stage may propagate to the editing process. Under strong manipulation strengths, minor identity drift may also occur due to amplified latent perturbations.

To better understand these limitations, we further analyze failure cases across 16 representative target attributes, generating 100 edited samples per attribute (1,600 samples in total). A result is considered a failure when the edited region does not match the target semantics or exhibits noticeable artifacts or identity inconsistency. As shown in Figure C.16, geometry- or boundary-sensitive attributes show relatively higher failure ratios, such as eyeglasses (18%) and curly hair (11%), whereas simpler attributes including smile, big eyes, and black clothes exhibit near-zero failure rates. These failures mainly occur in fine-grained local edits or structurally entangled attributes, where subtle semantic variations and correlated facial components increase manipulation difficulty. Furthermore, the ablation study in Figure 10 shows that removing reconstruction

or cosine supervision significantly increases instability, highlighting the importance of accurate latent direction alignment for preserving identity and semantic consistency.

Overall, these observations suggest that while FaceEditor performs reliably for most semantic editing tasks, improving robustness under complex structural transformations and boundary-sensitive regions remains an open challenge. Future work will focus on enhancing fine-grained editing stability in richer semantic contexts and extending the framework to more general image editing tasks beyond the facial domain. In addition, broader evaluations across diverse image distributions and careful consideration of potential ethical risks will be important for responsible deployment.

## References


[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., Learning transferable visual models from natural language supervision, in: International conference on machine learning, PMLR, 2021, pp. 8748–8763.

[2] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, et al., Oscar: Object-semantics aligned pre-training for vision-language tasks, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16, Springer, 2020, pp. 121–137.

[3] P. Xu, B. Jiang, X. Hu, D. Luo, Q. He, J. Zhang, C. Wang, Y. Wu, C. Ling, B. Wang, Unveil inversion and invariance in flow transformer for versatile image editing, in: Proceedings of the Computer Vision and Pattern Recognition Conference, 2025, pp. 28479–28489.

[4] A. B. Anees, A. C. Baykal, M. B. Kizil, D. Ceylan, E. Erdem, A. Erdem, Hypergan-clip: A unified framework for domain adaptation, image synthesis and manipulation, in: SIGGRAPH Asia 2024 Conference Papers, 2024, pp. 1–12.

[5] T. Wei, D. Chen, W. Zhou, J. Liao, Z. Tan, L. Yuan, W. Zhang, N. Yu, HairCLIP: Design your hair by text and reference image, in: Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 18072–18081.

[6] H. Zhang, C. Wu, G. Cao, H. Wang, W. Cao, HyperEditor: Achieving both authenticity and cross-domain capability in image editing via hypernetworks, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 7051–7059.

[7] H. Zhao, L. Zhang, P. L. Rosin, Y.-K. Lai, Y. Wang, HairManip: High quality hair manipulation via hair element disentangling, Pattern Recognition 147 (2024) 110132.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, Advances in neural information processing systems 27 (2014).

[9] W. Xia, Y. Yang, J.-H. Xue, B. Wu, TediGAN: Text-guided diverse face image generation and manipulation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 2256–2265.

[10] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, D. Lischinski, StyleCLIP: Text-driven manipulation of stylegan imagery, in: Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 2085–2094.

[11] T. Wei, D. Chen, W. Zhou, J. Liao, W. Zhang, G. Hua, N. Yu, HairCLIPv2: Unifying hair editing via proxy feature blending, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 23589–23599.

[12] Y. Zhu, H. Liu, Y. Song, Z. Yuan, X. Han, C. Yuan, Q. Chen, J. Wang, One model to edit them all: Free-form text-driven image manipulation with semantic modulations, Advances in Neural Information Processing Systems 35 (2022) 25146–25159.

[13] Y. Lyu, T. Lin, F. Li, D. He, J. Dong, T. Tan, DeltaEdit: Exploring text-free training for text-driven image manipulation, in: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2023, pp. 6894–6903.

[14] Z. Zhang, J. Xie, Y. Lu, Z. Yang, Y. Yang, Enabling instructional image editing with in-context generation in large scale diffusion transformer, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems, 2025.

[15] Z. Yan, Y. Ma, C. Zou, W. Chen, Q. Chen, L. Zhang, Eedit: Rethinking the spatial and temporal redundancy for efficient image editing, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025, pp. 17474–17484.

[16] Y. Zeng, Y. Zhang, L. Jiachen, L. Shen, K. Deng, W. He, J. Wang, Hairdiffusion: Vivid multi-colored hair editing via latent diffusion, Advances in Neural Information Processing Systems 37 (2024) 5048–5073.

[17] Z. Yin, L.-H. Chen, L. Ni, X. Dai, Consistedit: Highly consistent and precise training-free visual editing, in: Proceedings of the SIGGRAPH Asia 2025 Conference Papers, 2025, pp. 1–11.

[18] G. Kim, T. Kwon, J. C. Ye, DiffusionCLIP: Text-guided diffusion models for robust image manipulation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 2426–2435.

[19] Z. Geng, B. Yang, T. Hang, C. Li, S. Gu, T. Zhang, J. Bao, Z. Zhang, H. Li, H. Hu, et al., Instructdiffusion: A generalist modeling interface for vision tasks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 12709–12720.

[20] P. Zhu, R. Abdal, J. Femiani, P. Wonka, Barbershop: GAN-based image compositing using segmentation masks, ACM Transactions on Graphics (TOG) 40 (2021) 1–13.

[21] M. Nikolaev, M. Kuznetsov, D. Vetrov, A. Alanov, HairFastGAN: Realistic and robust hair transfer with a fast encoder-based approach, Advances in Neural Information Processing Systems 37 (2024) 45600–45635.

[22] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, D. Cohen-Or, Designing an encoder for StyleGAN image manipulation, ACM Transactions on Graphics (TOG) 40 (2021) 1–14.

[23] T. Wei, D. Chen, W. Zhou, J. Liao, W. Zhang, L. Yuan, G. Hua, N. Yu, E2Style: Improve the efficiency and effectiveness of stylegan inversion, IEEE Transactions on Image Processing 31 (2022) 3267–3280.

[24] E. Richardson, Y. Alaluf, O. Patashnik, Y. Nitzan, Y. Azar, S. Shapiro, D. Cohen-Or, Encoding in style: a StyleGAN encoder for image-to-image translation, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2021, pp. 2287–2296.

[25] Y. Alaluf, O. Tov, R. Mokady, R. Gal, A. Bermano, HyperStyle: StyleGAN inversion with hypernetworks for real image editing, in: Proceedings of the IEEE/CVF conference on computer Vision and pattern recognition, 2022, pp. 18511–18521.

[26] T. Wei, Y. Zhou, D. Chen, X. Pan, Freeflux: Understanding and exploiting layer-specific roles in rope-based mmdit for versatile image editing, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025, pp. 16745–16754.

[27] Q. Wang, A. Cvejić, A. Eldesokey, P. Wonka, Editclip: Representation learning for image editing, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025, pp. 15960–15970.

[28] M. Huang, J. Cai, S. Jia, V. S. Lokhande, S. Lyu, Paralleledits: efficient multi-aspect text-driven image editing with attention grouping, Advances in Neural Information Processing Systems 37 (2024) 22569–22595.

[29] Z. Xu, T. Lin, H. Tang, F. Li, D. He, N. Sebe, R. Timofte, L. Van Gool, E. Ding, Predict, prevent, and evaluate: Disentangled text-driven image manipulation empowered by pre-trained vision-language model, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 18229–18238.

[30] T. Zhu, S. Zhang, J. Shao, Y. Tang, Kv-edit: Training-free image editing for precise background preservation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025, pp. 16607–16617.

[31] J. Sun, Q. Deng, Q. Li, M. Sun, Y. Liu, Z. Sun, AnyFace++: A unified framework for free-style text-to-face synthesis and manipulation, IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).

[32] T.-T. Nguyen, Q. Nguyen, K. Nguyen, A. Tran, C. Pham, Swiftedit: Lightning fast text-guided image editing via one-step diffusion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2025, pp. 21492–21501.

[33] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 4401–4410.

[34] M. Li, X. Gu, F. Chen, X. Xing, L. Wen, C. Chen, S. Zhu, Superedit: Rectifying and facilitating supervision for instruction-based image editing, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2025, pp. 19206–19215.

[35] P. Zhu, R. Abdal, Y. Qin, J. Femiani, P. Wonka, Improved StyleGAN embedding: Where are the good latents?, arXiv preprint arXiv:2012.09036 (2020).

[36] H. Liu, Y. Song, Q. Chen, Delving StyleGAN inversion for image editing: A foundation latent space viewpoint, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2023, pp. 10072–10082.

[37] C. Xiao, Q. Yang, X. Xu, J. Zhang, F. Zhou, C. Zhang, Where you edit is what you get: Text-guided image editing with region-based attention, Pattern Recognition 139 (2023) 109458.

[38] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, N. Sang, BiSeNet: Bilateral segmentation network for real-time semantic segmentation, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 325–341.