SIEDOB: Semantic Image Editing by Disentangling Object and Background

Wuyang Luo\textsuperscript{1} Su Yang\textsuperscript{1,\textsuperscript{2}} Xinjian Zhang\textsuperscript{1} Weishan Zhang\textsuperscript{2}
\textsuperscript{1}Shanghai Key Laboratory of Intelligent Information Processing, School of Computer Science, Fudan University
\{wyluo18, suyang\}@fudan.edu.cn
\textsuperscript{2}School of Computer Science and Technology, China University of Petroleum

Abstract

Semantic image editing provides users with a flexible tool to modify a given image guided by a corresponding segmentation map. In this task, the features of the foreground objects and the backgrounds are quite different. However, all previous methods handle backgrounds and objects as a whole using a monolithic model. Consequently, they remain limited in processing content-rich images and suffer from generating unrealistic objects and texture-inconsistent backgrounds. To address this issue, we propose a novel paradigm, Semantic Image Editing by Disentangling Object and Background (SIEDOB), the core idea of which is to explicitly leverages several heterogeneous subnetworks for objects and backgrounds. First, SIEDOB disassembles the edited input into background regions and instance-level objects. Then, we feed them into the dedicated generators. Finally, all synthesized parts are embedded in their original locations and utilize a fusion network to obtain a harmonized result. Moreover, to produce high-quality edited images, we propose some innovative designs, including Semantic-Aware Self-Propagation Module, Boundary-Anchored Patch Discriminator, and Style-Diversity Object Generator, and integrate them into SIEDOB. We conduct extensive experiments on Cityscapes and ADE20K-Room datasets and exhibit that our method remarkably outperforms the baselines, especially in synthesizing realistic and diverse objects and texture-consistent backgrounds. Code is available at https://github.com/WuyangLuo/SIEDOB.

1. Introduction

Semantic image editing has recently gained significant traction due to its diverse applications, including adding, altering, or removing objects and controllably inpainting, outpainting, or repainting images. Existing methods have made impressive progress benefiting from Generative Adversarial Networks (GAN) \cite{goodfellow2014generative, isola2017image} and have demonstrated promising results in relatively content-simple scenes, such as landscapes. However, they still suffer from inferior results for content-rich images with multiple discrepant objects, such as cityscapes or indoor rooms. This paper aims to improve editing performance in complex real-world scenes.

When image editing experts deal with a complex task, they decompose the input image into multiple independent and disparate elements and handle them with different skills. For example, when editing a photo with a congested road, they process the cars and pedestrians individually and then fill in the background. Inspired by this spirit, we propose a heterogeneous editing model, which mimics the experience of human experts and explicitly splits the compound content into two distinct components: Backgrounds and objects. The background has no regular shape and may span a large area. More importantly, the re-generated background’s texture and the existing area must be consistent. Foregrounds objects are class-specific and appear anywhere with various scales. Due to the significant differences between the two components, it is necessary to disentangle them from the input and process them with different networks.

Several recent works \cite{zhu2020sesame, wang2020asset, zhang2020spmpgan} have been dedicated to this task. SESAME \cite{zhu2020sesame} proposes a new pair of generator and discriminator based on the cGAN framework \cite{isola2017image}, which generates edited content through a single-shot inference. ASSET \cite{wang2020asset} builds a transformer-based architecture with a sparse attention mechanism and synthesizes
edited regions relying on a codebook generated by VQ-GAN [4]. SPMPGAN [28] proposes a coarse-to-fine generator equipped with style-preserved modulation layers to retain style consistency between edited regions and context. These methods effectively enhance the ability of a monolithic model to handle the entire input. However, they deal with backgrounds and objects equally using the same generator, which causes them to produce inferior results. Figure 1 demonstrates an example. To tackle this limitation, we present a novel framework, Semantic Image Editing by Disentangling Object and Background (SIEDOB), to generate backgrounds and foreground objects separately, which can achieve two goals: Synthesizing texture-consistent backgrounds and generating photorealistic and diverse objects.

SIEDOB first disassembles the edited image into background regions and instance-level objects, then employs different generators to synthesize the corresponding content, and finally aggregates generated objects with backgrounds via a fusion network. In this way, we decouple our task into several more feasible subtasks and handle different components using dedicated sub-models.

Generating texture-consistent backgrounds with known context is not trivial since backgrounds may cross a sizeable spatial area in an image and have no regular shape. We propose a pair of boosted generator and discriminator to address this problem. Specifically, we propose Semantic-Aware Self-Propagation Module (SASPM) to help the generator efficiently transfer the semantic-wise features of known regions to generated regions. Moreover, to further enhance texture consistency, we design a Boundary-Anchored Patch Discriminator to force the generator to pay more attention to local textures of editing fringe.

In our task, depending on the scenario, the edited object may be requested to be inpainted or generated. If an object is partially visible, it undergoes a lightweight inpainting network. Otherwise, we employ a Style-Diversity Object Generator to obtain multi-modal results based on a style bank. After generating the backgrounds and all objects, we re-integrate them into a whole. However, the separate generation may lead to sudden boundaries and dissonance between objects and surroundings. To tackle this problem, we utilize a simple fusion network that predicts the residual value for each pixel to harmonize outputs.

Our main contributions can be summarized as follows:

- We propose a Style-Diversity Object Generator that can generate diverse and realistic object images from masks.
- Extensive experiments on two benchmarks indicate that our method can generate texture-consistent results and deal with crowded objects well.

2. Related Works

2.1. Semantic Image Editing

Semantic image editing refers to modifying a given image by manipulating a corresponding segmentation map and keeping the context coherent. This approach provides powerful editing capabilities, such as adding and removing objects, modifying shapes, and re-painting backgrounds. Compared to several related tasks, including semantic image synthesis [14, 33, 40, 42, 51] and image inpainting [13, 30, 45, 46, 48], semantic image editing has not been fully exploited due to its difficulty in generating realistic objects and texture-consistent backgrounds simultaneously. A few recent works are devoted to semantic image editing [10, 23, 28, 32]. HIM [10] is an early exploration that only operates on a single object using a two-stage network. SESAME [32] introduces a new pair of generators and discriminators to improve the quality of the generated results. Moreover, SESAME adopts a more flexible workflow that can respond to various image editing requirements of users. ASSET [23] proposes a novel transformer-based approach modeling long-range dependencies, enabling high-resolution image editing. SPMPGAN [28] presents a style-preserved modulation technique and builds a progressive architecture that solves the style inconsistency problem in semantic image editing. These methods all consider semantic image editing as a global generation problem. Thus, they use a single model to process all elements of the edited image equally. In this work, we propose a decoupled framework to handle objects and backgrounds separately using heterogeneous generators, demonstrating more promising results.

2.2. Image Synthesis via Heterogeneous Generators

For a wide variety of image generation tasks, most approaches impose monolithic generators on the entire image, including unsupervised image generation [17, 18, 35], image-to-image translation [29, 33, 42, 51], image inpainting [24, 45, 46], and image editing [15, 26]. They share the same network structure and weights to generate all the content without specialized submodels for different semantic regions or classes. Other methods use different architectures or weights to generate different image elements to improve generation quality for different components. For example, they employ multiple local branches specialized
for different parts [6, 9, 11, 21], decoupling content and style [12, 20, 27], generating foreground and a single foreground object separately [39, 43], enhancing local generation [1, 37], or having separate modules for different semantic categories [22, 40]. However, these methods are developed for pure image generation or translation tasks where their input is noises, categories, segmentation maps, or full images. Therefore, they only need to synthesize simple but visually plausible results to fool the discriminator. Semantic image editing is a more challenging task, and its difficulty lies in keeping coherence between edited and known regions. In this paper, we propose a heterogeneous framework that utilizes different sub-models for foreground objects and backgrounds to improve their generation quality concurrently.

3. Proposed Method

Our goal is to edit a given image $I \in \mathbb{R}^{H \times W \times 3}$ guided by a user-provided segmentation map $S \in \mathbb{L}^{H \times W \times L}$ within an edited region defined by a mask map $M \in \mathbb{B}^{H \times W \times 1}$ whose value is 0 in the non-edited region and 1 in the edited region. Here $H$, $W$, and $L$ denote the height, width, and number of semantic categories, respectively. The edited region of $I$ is erased as input $I_e = I \times (1 - M)$. We create a background mask $M_B$ and all instance-level object masks $\{M^q_c\}$ in the edited region based on the segmentation map. Here, $c \in C$, $C$ is a pre-defined set of considered foreground categories, and $M^q_c$ is the $q$-th instance that falls into the category $c$. The background mask $M_B$ is composed of the remaining pixels. Thus, we can disassemble the input image $I_e$ to the background and objects according to $M_B$ and $\{M^q_c\}$. Then we generate the background and objects separately using the respective generators. Finally, we integrate all generated content via a fusion network. The workflow is depicted in Figure 2.

3.1. Background Generation

For image editing, the background usually needs to be partially generated to fill the space, such as sky or ground. Therefore, generating style-consistent texture patterns in edited regions is critical and challenging since the generated background and the known region coexist. In addition, an image’s background may be across a distant spatial distance and have no regular shape. Efficiently and accurately transferring feature styles from known to edited regions is vital for generating texture-consistent results. To this end, we propose a Semantic-Aware Self-Propagation Module (SASPM), which explicitly extracts semantic-wise feature codes from feature maps and propagates them to the corresponding regions at the same layer. Furthermore, we employ a Boundary-Anchored Patch Discriminator to enforce the generator to focus on local textures at the editing boundary.

The proposed background generator $G_B$ is shown in 3(c). We feed the disassembled background region into $G_B$ as the input: $I_B = I_e \times M_B$. $G_B$ follows an encoder-decoder structure. The encoder is composed of several successive GatedConv layers [46] with stride two, and the decoder contains corresponding GatedConv layers with up-sampling and several SASPM.

Semantic-Aware Self-Propagation Modules (SASPM)

SASPM aims at propagating features of known regions to edited regions in a semantic-aware manner. Taking the sky area as an example shown in Figure 3(a)(b), a SASPM is divided into two steps. First, we extract the semantic-aware feature code by computing the average of the known sky area. Then, the feature code is broadcast into the entire sky area according to the corresponding semantic mask. SASPM can transfer the feature codes of each semantic label at the same feature level without the constraint of the receptive field. Formally, let $F \in \mathbb{R}^{H \times W \times C}$ denote a feature...
map in front of a SASPM. \( H, W \) are the spatial dimension and \( C \) is the number of channels. Let \( U \in \mathbb{R}^{L \times C} \) represent \( L \) feature codes extracted from the known region, and the feature codes of non-background classes are set to zero. We generate a code map \( P \) based on the segmentation map \( S \in \mathbb{L}^{H \times W \times L} \) through a matrix multiplication to spatially broadcast the feature codes into the corresponding semantic region:

\[
P = S \otimes U
\]  

(1)

Thus, \( P \in \mathbb{R}^{H \times W \times C} \) is restored to the spatial dimension filled with the category-specific feature codes. Finally, we propagate \( P \) into the original \( F \) using a modulation operation following SPADE\(^{[33]}\). Specifically, we learn two parameters \( \gamma_P \in \mathbb{R}^{H \times W \times C} \) and \( \beta_P \in \mathbb{R}^{H \times W \times C} \) from \( P \) to modulate \( F \):

\[
F_P = \text{ReLU} (\gamma_P \odot \text{IN}(\text{Conv}(F)) \odot \beta_P)
\]

(2)

Where \( \text{Conv}(\cdot) \), \( \text{IN}(\cdot) \), and \( \text{ReLU}(\cdot) \) denote Convolutional Layer, Instance Normalization, and ReLU activation function, respectively. \( \odot \) and \( \oplus \) are element-wise multiplication and addition.

---

**Boundary-Anchored Patch Discriminator**

We employ two discriminators for our background generator: (1) A global discriminator used by previous works \cite{28}; (2) a novel Boundary-Anchored Patch Discriminator \( D_{BAP} \). We introduce \( D_{BAP} \) to enforce the generator to pay attention to local texture across the fringe of the edited area. We expect \( D_{BAP} \) can capture known and generated textures simultaneously to facilitate texture-consistent results. To this end, we randomly select several central points anchored on the boundary of the edited region to crop patches from real or generated images, as shown in Figure 4. Thus, the fake patches contain both real and fake textures, so \( D_{BAP} \) can distinguish between true and false through texture consistency.

The training objective of \( G_B \) is comprised of \( L_1 \) Distance Loss \( L_1 \), Perceptual Loss \( L_p \) \cite{16}, Global Adversarial Loss \( L_{GAN}^G \), and Local Patch Adversarial Loss \( L_{GAN}^L \):

\[
L_B = L_1 + \lambda_{PB} L_P + L_{GAN}^G + \lambda_{GAN}^L L_{GAN}^L
\]

(3)

Where \( L_1 \) and \( L_P \) force the generated results to be closer to the ground truth in RGB space and VGG space \cite{38}, \( L_{GAN}^G \) and \( L_{GAN}^L \) are applied to global images and local patches, respectively. All adversarial losses in this work are associated with a SNPatchGAN Discriminator \cite{46} using the hinge version \cite{2}.

---

**3.2. Object Generation**

Our task involves dealing with two scenarios: Inpainting or generating an object—the former aims at completing an object based on its visible part. The latter synthesizes an inexistent object from its mask with an arbitrary appearance. Unlike handling all objects at once, cropping each edited object and independently generating them allows
us to use aligned input images with the same size, which is beneficial for generating high-quality results. Specifically, we crop all objects \( \{\text{Crop}(I_e)^g\} \) from the edited region based on object masks \( \{M_e^g\} \). We leverage a two-way model to handle the different scenarios. One way is a lightweight inpainting network \( G_{OI} \) to complete objects. \( G_{OI} \) is a UNet-like [36] network and its training objective is:

\[
\mathcal{L}_{GI} = \mathcal{L}_s + \lambda_{IP} \mathcal{L}_p + \mathcal{L}_{GAN}.
\]

\( \mathcal{L}_s, \mathcal{L}_p, \) and \( \mathcal{L}_{GAN} \) are the same as the terms in Equation 6. The other way is the proposed **Style-Diversity Object Generator** \( G_{OG} \) for the multi-modal generation. Note that all categories of objects share \( G_{OI} \) and \( G_{OG} \).

### Style-Diversity Object Generator

\( G_{OG} \) generates a class-specific image from a one-hot segmentation map \( M_{sem} \in \mathbb{L}^{H \times W \times K} \) created by the corresponding object mask \( M_e^g \). \( K \) is the number of the predefined foreground class set. Motivated by disentanglement learning [12, 20, 27], we utilize style information to control the appearance of results for multi-modal generation. Specifically, we sample an image from the training set with the same class label and extract its style code \( W \in \mathbb{R}^{128} \) by a style encoder \( E_s \). Then we generate a style map \( M_{style} \in \mathbb{R}^{H \times W \times 128} \) via broadcasting the style code according to \( M_e^g \), as illustrated in Figure 5(a). At this point, we have acquired a semantic map \( M_{sem} \) and a style map \( M_{style} \). The next is how to decode them back to a style-controlled object image. To this end, we design a **Semantic-Style Normalization Module (SSNM)** to integrate the style and semantic information, as shown in Figure 5(b). We first inject the semantic information into the input feature maps \( F \in \mathbb{R}^{H \times W \times C} \):

\[
F_{sem} = \text{ReLU}(\gamma_c \odot \text{IN(Conv}(F)) + \beta_c)
\]

Where two normalization parameters, \( \gamma_c \in \mathbb{R}^{H \times W \times C} \) and \( \beta_c \in \mathbb{R}^{H \times W \times C} \) are learned from \( M_{sem} \). Similarly, the style information is injected by:

\[
F_{O} = \text{ReLU}(\gamma_s \odot \text{IN(Conv}(F_{sem})) + \beta_s)
\]

Where two normalization parameters \( \gamma_s \in \mathbb{R}^{H \times W \times C} \) and \( \beta_s \in \mathbb{R}^{H \times W \times C} \) are learned from \( M_{style} \).

During training, for each sample, we take the ground truth image \( I^{gt} \) and a randomly sampled image \( I^s \) as the style images to generate two results \( R^{gt} \) and \( R^s \). We employ the following objective function to optimize \( G_{OG} \):

\[
\mathcal{L}_{GO} = \mathcal{L}_1 + \lambda_P \mathcal{L}_P + \mathcal{L}_{GAN} + \mathcal{L}_{SCC}
\]

Here, \( \mathcal{L}_1 \) is only applied to \( R^{gt} \). \( \mathcal{L}_P, \mathcal{L}_{GAN}, \) and \( \mathcal{L}_{SCC} \) are imposed on all results. We introduce **Style Cycle-Consistency Loss** \( \mathcal{L}_{SCC} \) to force the generator to produce style-consistent results with style images:

\[
\mathcal{L}_{SCC} = 1 - \cos \left( \frac{E_S'(R)}{\|E_S'(R)\|_2} \cdot \frac{E_S'(I^{style})}{\|E_S'(I^{style})\|_2} \right)
\]

Where \( I^{style} \) is the style image and \( R \) is the generated result. \( E_s'(\cdot) \) is a style encoder with the same structure as \( E_s \). \( \cos(\cdot, \cdot) \) represents the cosine similarity of two vectors.

Previous image translation methods [12, 20] achieve multi-modal generation by adding Gaussian noise during
inferne. We find that noise leads to low-quality results and limited diversity. To solve this problem, we replace the noise with a style bank. Precisely, once training is done, we extract all style codes from the training set by the trained $E_x$ and save them for building a class-aware style bank. During inference, we randomly sample different style codes from the style bank to generate different results. It is worth mentioning that our approach does not increase any additional computational overhead compared to adding noise.

### 3.3. Fusion Network

After generating all backgrounds and objects, we embed them into their original positions to obtain a composited image. However, this preliminary result may be discordant since each object is generated independently without accounting for context. We employ a fusion network $F$ to harmonize the objects with the surroundings, as shown in Figure 6. $F$ learns an offset value for each pixel via a skip connection [7, 36] to produce the final results. We use Perceptual Loss and Adversarial Loss to optimize $F$:

$$L_F = \lambda_{PF} L_P + L_{GAN}$$  \hspace{1cm} (8)

### 4. Experiments

**Datasets.** We conduct experiments on two datasets, Cityscapes and ADE20K-Room, used in [28, 32]. Cityscapes [3] contains complex street scene images in German cities. ADE20K-Room is a subset of ADE20K [50] consisting of indoor scene images. We train and test our model at $256 \times 256$ resolution.

**Implementation Details.** We use free-form, extension, and outpainting masks following [28]. Similar to the previous work [22], we choose those frequently-appearing categories as the foreground class set, (car, person) for Cityscapes and (bed, chest, lamp, chair, table) for ADE20K-Room. All sub-networks are independently trained using ADAM optimizers [19] for both the generator and the discriminators with momentum $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and the learning rates for the generator and the discriminators are set to 0.0001 and 0.0004, respectively. For training $D_{BAP}$, we randomly crop 4 square patches with the size ranging from $96 \times 96$ to $160 \times 160$. We set balance coefficients $\lambda_{PB} = \lambda_{PI} = \lambda_{PO} = \lambda_{PF} = 10$ and $\lambda_{GAN} = 0.2$. The proposed method is implemented with Pytorch 1.10 [34], and all experiments are performed on a single NVIDIA RTX 3090 GPU.

**Baselines.** We compare the proposed methods with the state-of-the-art semantic image editing methods, including HIM [10], SESAME [32], ASSET [23], SPMPGAN [28]. We also introduce two additional methods [40, 48] as baselines to investigate related work more widely. LGGAN [40] learns a map from segmentation maps to photo-realistic images using global image-level and class-specific generators. Co-Mod [48] demonstrates impressive results in semantic image generation and image inpainting. We change their input to be the same as our task.

### 4.1. Qualitative evaluation

We demonstrate a visual comparison of the competing methods, as shown in Figure 7. We observe that the results produced by the proposed $SIEDOB$ are visually better than existing methods in two aspects: (1) Our method can well handle dense objects overlapping each other, e.g., in the second and third rows. However, other methods produce distorted and inseparable results. (2) Our method generates texture-consistent content between edited regions and known regions both on background and objects, e.g., the red cars in the first row and the curtain in the fourth row. The performance improvement benefits from our disentangling generation strategy and carefully designed sub-networks.

### 4.2. Quantitative evaluation

Table 1 lists the quantitative results with three different mask types. Following previous works, we use three metrics: FID [8], LPIPS [47], and mean Intersection-over-Union (mIoU). FID is introduced to assess the fidelity of the
also indicate the superiority and DRN-D-105 for Cityscapes. Our method outperforms the other methods under different metrics and mask settings.

<table>
<thead>
<tr>
<th>M. Method</th>
<th>Cityscapes</th>
<th>ADE20K-Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGGAN</td>
<td>15.29 0.094 58.62 24.74 0.097 27.68</td>
<td>19.46 0.159 59.63 41.52 0.288 27.85</td>
</tr>
<tr>
<td>Co-Mod</td>
<td>15.88 0.097 56.50 27.37 0.111 27.52</td>
<td>29.27 0.188 56.44 38.61 0.231 27.13</td>
</tr>
<tr>
<td>HIM</td>
<td>15.58 0.093 58.99 28.64 0.133 28.04</td>
<td>25.20 0.180 58.91 40.69 0.239 27.61</td>
</tr>
<tr>
<td>E. SPMPGAN</td>
<td>12.89 0.082 58.88 21.73 0.101 27.50</td>
<td>20.30 0.168 59.08 36.43 0.211 27.62</td>
</tr>
<tr>
<td>ASSET</td>
<td>13.67 0.098 58.12 30.63 0.126 26.02</td>
<td>21.99 0.186 58.01 38.17 0.261 27.03</td>
</tr>
<tr>
<td>SPMPGAN</td>
<td>11.90 0.084 58.80 18.83 0.090 28.22</td>
<td>19.46 0.167 59.10 32.92 0.199 27.73</td>
</tr>
<tr>
<td>SIEDOB</td>
<td>11.07 0.077 59.41 41.07 0.213 27.91</td>
<td>19.20 0.159 59.63 31.66 0.191 29.52</td>
</tr>
</tbody>
</table>

Figure 8. Visual results of addition objects.

Figure 9. Multi-modal generation results.

Table 1. Quantitative comparison with other methods. M., E., and O. represent Mask Type, Free-Form Mask, Extension Mask, and Outpainting Mask, respectively. (↑: Higher is better; ↓: Lower is better)

4.3. Addition Object

Adding a new object to the given image is an essential ability of semantic image editing. The visual results are shown in Figure 8. In this experiment, we randomly select an instance of each test image and create a rectangular mask encircling the object. The proposed method can generate realistic objects with crisp edges. Adding an object to the edited area that does not exist originally should have diverse results. Some previous methods [23,48] have multi-modal generation capabilities. We sample two results from all methods, as shown in Figure 9. Our method achieves a higher diversity than other models because the previous methods only exert a global influence on the generation process; in contrast, our model independently generates each object using different class-aware style codes. The quantitative results listed in Table 2 also indicate the superiority of our method. Here, the Diversity score (Div) is calculated as the average LPIPS distance between pairs of randomly sampled results for all test images, as also done in [25,49]. Moreover, our model can hallucinate images that do not exist in the dataset’s distribution, as shown in Figure 10.

4.4. Ablation Study

Effectiveness of SASPM and Boundary-Anchored Patch Discriminator. We investigate the contributions of the proposed SASPM and $D_{BAP}$ to the generation of texture-consistent background. To exclude the interference of the foreground, we create mask maps that only include back-
ground regions in this study. We remove all SASPM ("w/o SASPM") or $D_{BAP}$ ("w/o $D_{BAP}$") from our model as two baselines. The visual results are shown in Figure 11 and prove that SASPM and $D_{BAP}$ can help the background generator effectively alleviate texture inconsistencies. Vanilla convolutional layers have limited receptive fields and no ability to locate where edited or known regions are. SASPM explicitly transfers features from known regions to edited regions with the same semantic labels across distant spatial distances. $D_{BAP}$ forces the generator to pay more attention to local textures on editing boundaries rather than just global images. Quantitative results are listed in Table 3.

Effectiveness of Style-Diversity Object Generator. To verify the benefit of $L_{SCC}$ and the style bank, we set up two baselines: (1) Dropping out $L_{SCC}$ ("w/o $L_{SCC}$"). (2) Replacing the style codes with noise vectors ("w Noise"). Figure 12 shows some visual results. If we drop out $L_{SCC}$, the generated image’s style may not be consistent with the style image. If we produce multi-modal results by injecting noise, it leads to a loss of fidelity and diversity. This is because Gaussian noise is semantics-free, while style code is category-specific and provides controllability. The quantitative results are listed in Table 4.

Effectiveness of Separate Generation. We remove the object generator and the fusion network but keep other components unchanged to obtain a baseline (w/o Separation) and increase its channel number for a fair comparison. Qualitative and quantitative comparisons can be found in Figure 13 and Table 5, respectively. In this experiment, we employ the masks used in object addition. The visual comparison shows that the separate generation paradigm significantly improves performance for objects with complicated surroundings.

Effectiveness of Fusion Network. Fusion network can eliminate abrupt boundaries between generated objects and backgrounds. Removing the fusion network (w/o Fusion) or not using the skip connection (w/o Skip) will degrade the performance, which are qualitatively and quantitatively demonstrated in Figure 13 and Table 5.

5. Conclusion and Limitations

This paper proposes a novel paradigm for semantic image editing, named SIEDOB. Its core idea is that since the characteristics of foreground objects and backgrounds are dramatically different, we design a heterogeneous model to handle them separately. To this end, we present SASPM and Boundary-Anchored Patch Discriminator to facilitate the generation of texture-consistent backgrounds and employ a Style-Diversity Object Generator to produce high-quality and diverse objects. Extensive experimental results demonstrate the superiority of our method. However, our method has some limitations. Some foreground categories are very rare in the dataset, so we cannot obtain enough data for training. In addition, object generation is hard to produce satisfactory results when an object is with an extreme pose or large-scale occlusion.

Acknowledgement This work is supported by State Grid Corporation of China (Grant No. 5500-202011091A-0-0-000).
References


[26] Hongyu Liu, Ziyu Wan, Wei Huang, Yibing Song, Xintong Han, Jing Liao, Bin Jiang, and Wei Liu. Deflocnet: Deep image editing via flexible low-level controls. In Proceedings of


[51] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive nor-