

DeFLOCNet: Deep Image Editing via Flexible Low-level Controls

Hongyu Liu¹ Ziyu Wan² Wei Huang³ Yibing Song^{4*} Xintong Han¹
 Jing Liao² Bing Jiang³ Wei Liu⁵
¹Huya Inc ²City University of Hong Kong ³Hunan University
⁴Tencent AI Lab ⁵Tencent Data Platform

{liuhongyu1, hanxintong}@huya.com ziyuwan2-c@my.cityu.edu.hk
 yibingsong.cv@gmail.com wl2223@columbia.edu

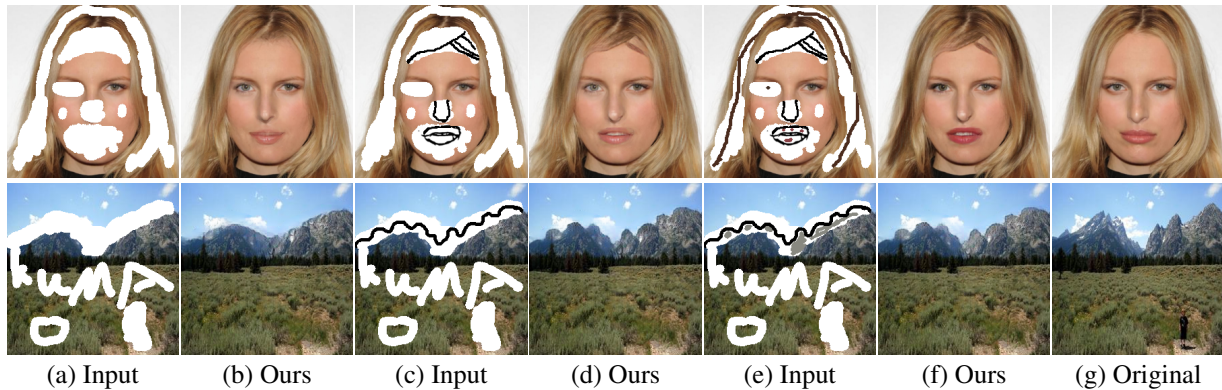


Figure 1. Diversified image editing results under free-form input configurations. There is a face or a natural image with arbitrary hole regions. Without any user inputs shown in (a), our DeFLOCNet automatically fills hole regions shown in (b), which is the same as the image inpainting scenario. Given additional low-level controls (e.g., coarse sketch lines in (c), both lines and colors in (e)), our DeFLOCNet injects these controls directly into structure generation blocks for both user-intended and visually pleasant content creations in (d) and (f).

Abstract

User-intended visual content fills the hole regions of an input image in the image editing scenario. The coarse low-level inputs, which typically consist of sparse sketch lines and color dots, convey user intentions for content creation (i.e., free-form editing). While existing methods combine an input image and these low-level controls for CNN inputs, the corresponding feature representations are not sufficient to convey user intentions, leading to unfaithfully generated content. In this paper, we propose DeFLOCNet which relies on a deep encoder-decoder CNN to retain the guidance of these controls in the deep feature representations. In each skip-connection layer, we design a structure generation block. Instead of attaching low-level controls to an input image, we inject these controls directly into each structure generation block for sketch line refinement and color propagation in the CNN feature space. We then concatenate the modulated features with the original decoder features for structure generation. Meanwhile, DeFLOCNet involves

another decoder branch for texture generation and detail enhancement. Both structures and textures are rendered in the decoder, leading to user-intended editing results. Experiments on benchmarks demonstrate that DeFLOCNet effectively transforms different user intentions to create visually pleasing content.

1. Introduction

The investigation on image editing is growing as it reduces significant manual efforts during image content generation. Benefiting from the realistic image representations brought by convolutional neural networks (CNNs), image editing is able to create meaningful while visually pleasant content. As shown in Fig. 1, users can draw arbitrary holes in a natural image as inputs to indicate the regions to be edited. If there are no further inputs given as shown in (a), image editing degenerates to image inpainting, where CNNs automatically fill hole regions by producing coherent image content as shown in (b). If there are additional inputs from users (e.g., lines in (c) and both lines and colors in (e)), CNNs will create meaningful content accordingly

*Y. Song is the corresponding author. The results and code are available at <https://github.com/KumapowerLIU/DeFLOCNet>.

while maintaining visual pleasantness. Deep image editing provides flexibility for users to generate diversified content, which can be widely applied in the areas of data enhancement, occlusion removal, and privacy protections.

The flexibility of user controls and the quality of user-intended content generation are challenging to achieve simultaneously in practice. The main difficulty resides on how to transform flexible controls into user-intended content. Existing attempts utilize high-level inputs (*e.g.*, semantic parsing map [8], attributes [22], latent code [1], language [2], and visual context [21]) for semantic content generation, but flexibility hinges on the predefined semantics.

On the other hand, utilizing coarse low-level controls (*i.e.*, sketch lines and colors) makes the editing more interactive and flexible. And in this paper, we mainly focus on incorporating such user inputs for image editing, in which we observe two main challenges: (1) Most prior investigations [33, 10, 23] simply combine an input image and low-level controls together in the image level for CNN inputs. The guidance from these low-level inputs gradually diminishes in the CNN feature space, weakening their influence on generating user-intended contents. Fig. 6 (c)-(f) show such examples where facial components are not effectively produced, (2) Since users only provide sparse color strokes to control the generated colors, the model needs to propagate these spatially sparse signals to the desired regions guided by sketches (*i.e.*, colors should fill in the regions indicated by the sketches and not be wrongly rendered across sketch lines) as illustrated in Figs. 5 and 7.

To resolve these issues, we propose DeFLOCNet (*i.e.*, Deep image editing via Flexible LO-level Control) to retain the guidance of low-level controls for reinforcing user intentions. Fig. 2 summarizes DeFLOCNet, which is built on a deep encoder-decoder for structure and texture generations on the hole regions. At the core of our contribution is a novel structure generation block (Fig. 3 and Sec. 3.1), which is plugged into each skip connection in the network. Low-level controls are directly injected into these blocks for sketch line generation and color propagation in the feature space. The structure features from these blocks are concatenated to the original decoder features accordingly for user-intended structure generation in the hole regions.

Moreover, we introduce another decoder for texture generation (Sec. 3.2). Each layer of the texture generation decoder is concatenated to the original decoder for texture enhancement. Thus, both structure and texture are effectively produced in the CNN feature space. They supplement original decoder features to bring coarse-to-fine user-intended guidance in the CNN feature space and output visually pleasing editing results. Experiments on the benchmark datasets demonstrate the effectiveness of our DeFLOCNet compared to state-of-the-art approaches.

2. Related Work

Deep Generative Models. The advancements in deep generative models [25, 27, 29] are inspired by generative adversarial learning [5, 26]. Instead of image generation from random noise, conditioned image generation from inputs activates a series of image translation work. In [9], a general framework is proposed to translate semantic labels to natural images. This framework is further improved by using a coarse-to-fine generator and a multi-scale discriminator [28]. Besides holistic image generation, subregion image generation (*i.e.*, image inpainting) receives heavy investigations [32, 16, 18]. In contrast to existing image-to-image generation frameworks, our free-form image editing is more flexible to transfer user intentions (*i.e.*, monotonous sketch lines and color dots) into natural image content.

Image Editing. GANs have a lasting influence on image editing development. In [22], Invertible Conditional GANs are proposed to control high-level attributes of generated faces. Then, more effective editing is proposed in [24] by approximating a disentangled latent space. Semantic parsing maps are utilized in [8, 6, 3] as the intermediate representation for guided image editing, while natural language navigates editing in [2, 19]. Methods based on semantic guidance typically require an explicit correspondence between editing content and semantic guidance. As the semantic guidance is usually fixed with limited options, the editing is thus not flexible (*e.g.*, color and sketch controls). To improve the input flexibility, SC-FEGAN [10] proposes to directly combine sketch lines and colors as inputs and send them together with an input image to CNN. Gated convolution is proposed in [33] for flexibility improvement. As these methods attach user controls directly to input images for CNN input, the influence of user controls diminishes gradually. As a result, limited editing scenarios are supported by these methods (*e.g.*, facial component editing). Different from existing approaches, we inject low-level controls in the skip connection layers of an encoder-decoder with our structure generation block to gradually reinforce user intentions in a coarse-to-fine manner.

3. DeFLOCNet

Fig. 2 shows an overview of our DeFLOCNet built on top of an encoder-decoder CNN model. The input to the encoder is an image with arbitrary hole regions. Low-level controls, represented by sketch lines and color dots, are sent to the structure generation blocks (SGB) set on the skip-connection layers (Sec. 3.1). Meanwhile, we propose another decoder named texture generation branch (TGB) (Sec. 3.2). The features from SGB and TGB are fused with the original decoder features hierarchically for output image generation.

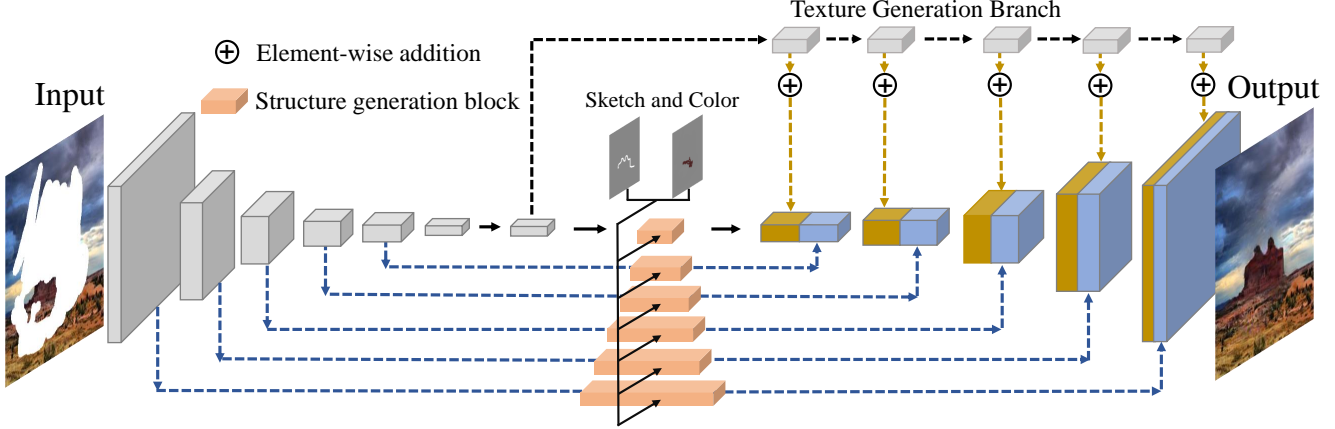


Figure 2. An overview of DeFLOCNet. We set each structure generation block within each skip connection layer. The block size corresponds to the level of skip connection layers. Low-level (*i.e.*, free-form) controls are injected into these blocks to modulate encoder features from coarse to fine. The modulated features represent user intentions and supplement decoder features together with texture generation features for output image generation.

Motivation. A vanilla encoder-decoder CNN may be prevalent for image generation while it is insufficient to recover missing content in the hole regions which are almost empty with sparse low-level controls. This is due to the limited ability of the encoder features to represent both natural image contents and free-form controls. To improve feature representations, we only send images to the encoder while injecting controls multiple times into all the skip connection layers via SGB. Consequently, user intentions are reinforced continuously via feature modulations. The reinforced features, together with the texture generation features, supplement the original decoder features in a coarse-to-fine fashion to generate both user-intended and visually pleasant visual content.

3.1. Structure Generation

We inject low-level controls into SGBs to modulate features of user intention reinforcement. Fig. 3 shows the architecture of one SGB, which consists of three branches for progressive sketch line generation, color propagation, and feature fusion, respectively. The size of one SGB increases when it is integrated into a shallower encoder layer, since the shallower is close to the image level and we need stronger low level to guide the feature generation. The sketch line generation branch repeatedly injects and refines the sketch generation, avoiding the user control diminishing phenomenon. Then, the features of sketch are utilized in the color propagation branch to regularize the color to fill in the desired regions. Finally, the fusion branch injects the sketch and color features into the original feature to produce output editing results.

Control injection. We first introduce a control injection operation, which is a basic building block in our SGB.

The control operation follows [20]. Specifically, we denote $F^{\text{in}} \in \mathbb{R}^{H \times W \times C}$ as the input feature map and L as the information we want to inject into F^{in} . Suppose the injection operation is $\mathcal{I}(\cdot)$, and the element in the injected feature $F_{x,y,c}^{\text{out}} = \mathcal{I}(F_{x,y,c}^{\text{in}}, L)$ can be obtained by:

$$\mathcal{I}(F_{x,y,c}^{\text{in}}, L) = \gamma_{x,y,c}(L) \cdot \frac{F_{x,y,c}^{\text{in}} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + \beta_{x,y,c}(L), \quad (1)$$

where $\gamma_{x,y,c}(L)$ and $\beta_{x,y,c}(L)$ are two variables controlling the influence from L in element-wise precision, $\mu_c = \frac{1}{H \times W} \sum_{x=1}^H \sum_{y=1}^W F_{x,y,c}^{\text{in}}$, and $\sigma_c = \sqrt{\frac{1}{H \times W} \sum_{x=1}^H \sum_{y=1}^W (F_{x,y,c}^{\text{in}} - \mu_c)^2}$. In this paper, we use two convolutional layers to generate $\gamma_{x,y,c}(L)$ and $\beta_{x,y,c}(L)$ at each element location. As a result, low-level controls are mapped into the feature space to correlate to the input feature maps.

Sketch line generation. Given an input feature F^{in} , the sketch line generation branch first performs an element-wise average along the channel dimension to produce a single-channel feature map $F_{\text{avg}}^{\text{in}}$. We denote the sketch image as S , a random noise image as N , and a mask image containing a hole region as M . The output feature after control injection can be written as:

$$F_s = \mathcal{I}(F_{\text{avg}}^{\text{in}}, S \oplus (N \odot M)), \quad (2)$$

where F_s is the injected feature, \oplus is the concatenation operator, and \odot is the element-wise multiplication operator. In practice, a single injection does not generate recovered sketch lines completely. We use several injections in the sketch line generation branch to progressively refine sketch

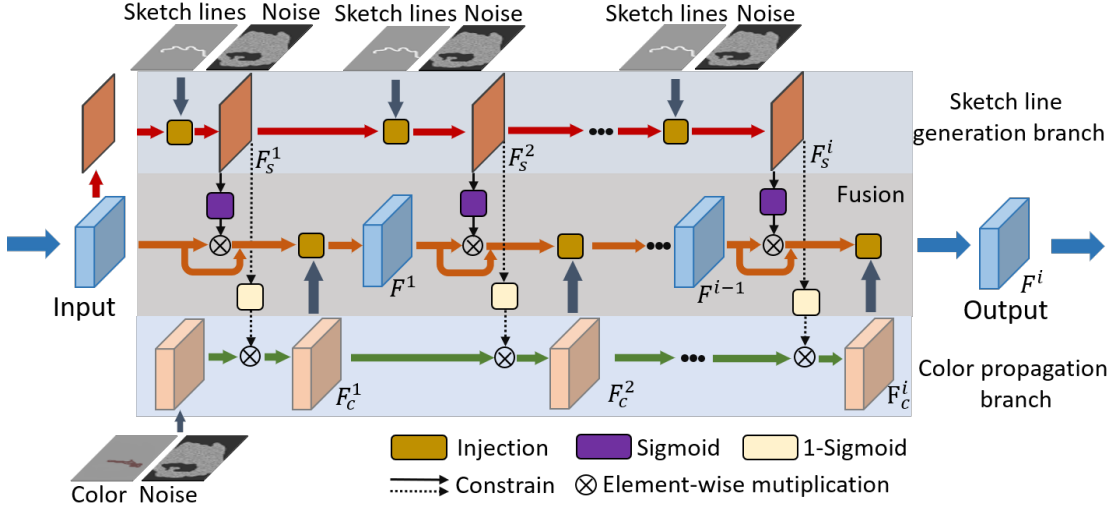


Figure 3. The architecture of a structure generation block where $i \in [1, 2, \dots, 6]$ denotes its size. The size of an SGB increases (*i.e.*, i increases) when it is integrated into a shallower encoder layer. An SGB consists of one sketch line generation branch, one color propagation branch, and a fusion branch. The sketch lines are gradually refined to guide the color propagation. These features are fused with the input image features for content generation in the hole region.

lines. The injection in the i -th skip connection is:

$$F_s^i = \mathcal{I}(\text{Conv}(F_s^{i-1}), S \oplus (N \odot M)), \quad (3)$$

where the output features from the previous injection F_s^{i-1} are passed through a convolutional layer for the current injection input. During training, we use the ground truth sketch lines extracted from the original images. When editing images, we adopt user inputs for sketch line refinement.

Color propagation. We propose a color propagation branch in parallel to the sketch generation branch. In order to guide the color propagation via sketch lines, we use injected features F_s^i from the sketch line generation branch. The guiding process can be written as:

$$F_c^i = (1 - \sigma(F_s^{i-1})) \otimes \text{Conv}(F_c^{i-1}), \quad (4)$$

where F_c^i is the color features guided by $F_s^{(i-1)}$ and σ is the sigmoid activation function.

Fig. 4 illustrates how color propagates under sketch guidance. In (a) and (b), the sketch lines in gray are not recovered well and the blue color tends to diffuse in all directions. As the sketch lines are gradually refined to complete contours as shown in (c)-(e), the blue color does not penetrate the contour lines via the consecutive σ and $1 - \sigma$ operations. Finally, blue color propagates along the contour lines without penetration as shown in (f).

Fusion. The features from the sketch and color branches are fused together via the injection operation as follows:

$$F^i = \mathcal{I}(F^{i-1} \otimes (\sigma(F_s^{i-1}) + \mathbf{1}), F_c^{i-1}), \quad (5)$$

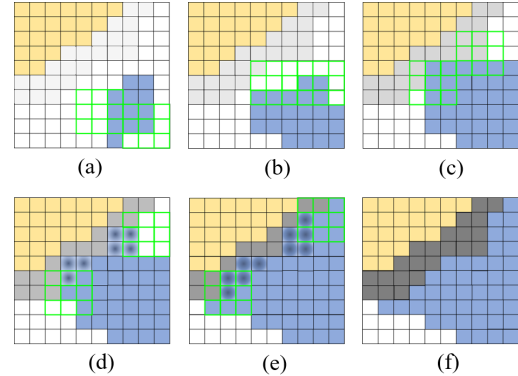


Figure 4. Sketch line refinement guides color propagation. The green window indicates the regular convolution operation, which is denoted by Conv in Eq. 4. The blue color diffuses along all directions in (a)-(b) when sketch lines (gray lines) are initially refined. Color propagation is weakened via $1 - \sigma$ around the contour lines that are gradually completed as shown in (c)-(e). The blue dots indicate the weakened elements along the lines. The propagation result is shown in (f) where blue color propagation follows contour lines.

where F^i is the fused feature. We set different numbers of injection operations in the fusion branch for each scale, respectively. For the skip-connection from the initial encoder layer where features are in large resolution, we employ 6 injection operations in the fusion branch. We gradually decrease this number to 1 on the skip connection layers from the last encoder layer.

3.2. Texture Generation

We use SGB to reinforce user intentions during hole filling in the CNN feature space. The modulated features represent structure content while texture representation is limited. This is partially because low-level controls injected into the skip-connection layers do not contain sufficient texture guidance. Meanwhile, the sketch lines attend the encoder features to structure content rather than textures.

As encoder features do not relate to low-level controls, we propose a texture generation branch that takes the features from the last encoder layer as input. Fig. 2 shows how TGB is integrated into the current pipeline. The architecture of TGB is the same as that of the original decoder. We add the feature maps from each layer of TGB to the corresponding decoder features for texture generation. TGB supplements decoder features via residual aggregations. As structure features are learned via SGB, TGB will focus on the features representing region details. The enriched decoder features are then concatenated with the features from SGBs for output generation where there are both structures and textures in the hole regions.

3.3. Objective Function

We utilize several objective loss functions to train DeFLOCNet in an end-to-end fashion. These functions include pixel reconstruction loss, perceptual loss [11], style loss [4], relativistic average LS adversarial loss [12], and total variation loss [16]. During training, we extract the sketch lines and color in the hole regions. We denote I_{out} as the output result and I_{gt} as the ground truth ($I_{\text{out}}, I_{\text{gt}} \in \mathbb{R}^{H \times W \times 3}$). The loss terms can be written as follows:

Pixel reconstruction loss. We measure the pixel-wise difference between I_{gt} and I_{out} as:

$$L_{\text{re}} = \|I_{\text{out}} - I_{\text{gt}}\|_1, \quad (6)$$

Perceptual loss. We consider high-level feature representation and human perception to utilize the perceptual loss, which is based on the ImageNet-pretrained VGG-16 backbone. The perceptual loss can be written as:

$$L_{\text{prec}} = \sum_i \frac{1}{N_i} \|F_i(I_{\text{out}}) - F_i(I_{\text{gt}})\|_1, \quad (7)$$

where F_i is the feature map of the i -th layer of the VGG-16 backbone and, N_i is the number of elements in F_i . In our work, F_i corresponds to the activation maps from layers ReLu1_1, ReLu2_1, ReLu3_1, ReLu4_1, and ReLu5_1.

Style loss. The transposed convolutional layers of the decoder will bring checkerboard effect [16], which can be mitigated by the style loss. Suppose the size of feature map F_i

is $C_i \times H_i \times W_i$. We write the style loss as:

$$L_{\text{style}} = \sum_i \frac{1}{M_i} \|G_i^F(I_{\text{out}}) - G_i^F(I_{\text{gt}})\|_1, \quad (8)$$

where G_i^F is a $C_i \times C_i$ Gram matrix computed based on the feature maps, and M_i is the number of elements in G_i^F . These feature maps are the same as those used in the perceptual loss as illustrated above.

Relativistic average LS adversarial loss. We utilize global and local discriminators for perception enhancement. The relativistic average LS adversarial loss is adopted for our discriminators, which can be written as:

$$L_{\text{adv}} = -\mathbb{E}_{x_r} [\log(1 - D_{ra}(x_r, x_f))] - \mathbb{E}_{x_f} [\log(D_{ra}(x_f, x_r))], \quad (9)$$

where $D_{ra}(x_r, x_f) = \text{sigmoid}(C(x_r) - \mathbb{E}_{x_f}[C(x_f)])$ and $C(\cdot)$ indicates the local or global discriminator, and real and fake data pairs (x_r, x_f) are sampled from I_{gt} and I_{out} .

Total variation loss. This loss is set to add the smoothness penalty on the generated regions, which is defined as:

$$L_{\text{tv}} = \sum_{(i,j) \in R, (i,j+1) \in R} \frac{\|I_{\text{out}}^{i,j+1} - I_{\text{out}}^{i,j}\|_1}{N} + \sum_{(i,j) \in R, (i+1,j) \in R} \frac{\|I_{\text{out}}^{i+1,j} - I_{\text{out}}^{i,j}\|_1}{N}, \quad (10)$$

where N is the number of elements in I_{out} , and R denotes the hole regions.

Total losses. The whole objective function of DeFLOCNet can be written as:

$$L_{\text{Total}} = \lambda_1 \cdot L_{\text{re}} + \lambda_2 \cdot L_{\text{prec}} + \lambda_3 \cdot L_{\text{style}} + \lambda_4 \cdot L_{\text{tv}} + \lambda_5 \cdot L_{\text{adv}}, \quad (11)$$

where $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ and λ_5 are the scalars controlling the influence of each loss term. We empirically set $\lambda_1 = 1, \lambda_2 = 0.05, \lambda_3 = 250, \lambda_4 = 0.1$ and $\lambda_5 = 0.1$.

3.4. Visualizations

The feature representations of input sketch lines are gradually enriched to become those of contours in one SGB. The enriched lines guide the color propagation process for edge-preserving feature modulation. To validate this effect, we visualize the feature maps from the fusion branch of one SGB set in the coarse level. Following the visualization techniques in [17], we visualize the 6 injection operations in the fusion branch. Specifically, we use a 1×1 convolutional layer to map each CNN feature F^i ($i \in [1, 2, \dots, 6]$) to one color image, and another 1×1 convolutional layer

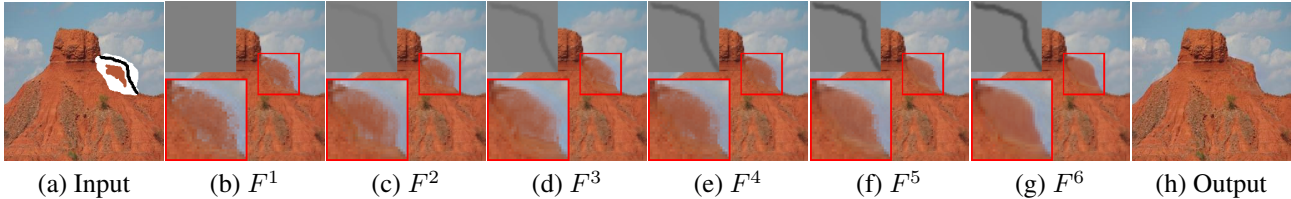


Figure 5. Feature map visualizations. We map F^i to color images via a 1×1 convolutional layer and show F_s^i accordingly at the top left corner. The color propagates according to the sketch line refinement, which validates the illustration in Fig. 4.

to map each CNN feature F_s^i to one grayscale image. The weights of the mapping convolutions are learned. The content shown in the images indicates the corresponding feature representations in SGB.

Fig. 5 shows the visualization results. The input image is shown in (a) where the low-level controls are sent to SGBs. The visualization of F^i is shown in (b)-(g) where the features F_s^i from the sketch generation branch are shown at the top left corner, respectively. We observe that during initial sketch line generation shown in (b)-(c), color propagates in all directions. When sketch lines are gradually completed as shown in (d)-(g), color propagates along these lines to formulate a clear boundary. These feature maps F^i are then concatenated to the original decoder for image structure generation shown in (h). The visualization of color propagation is similar to that in Fig. 4, where the $1 - \sigma$ operation is effective to prevent the color diffusions.

4. Experiments

We evaluate on a natural image dataset Places2 [34] and a face image dataset CelebA-HQ [14]. During training, we follow PConv [16] and HED [30] to create hole regions and input sketch lines, respectively. The color inputs for face images are from GFC [15] and for natural images are from RTV [31]. During training, we choose arbitrary hole regions and low-level controls. Adam optimizer [13] is adopted with a learning rate of 2×10^{-4} . The training epochs for CelebA-HQ and Place2 datasets are 120 and 40, respectively. The resolution of synthesized images is 256×256 . All the experiments are conducted on one Nvidia 2080 Ti GPU. The various edges of training images ensure our method to handle diverse and deformed strokes.

4.1. State-of-the-art Comparisons

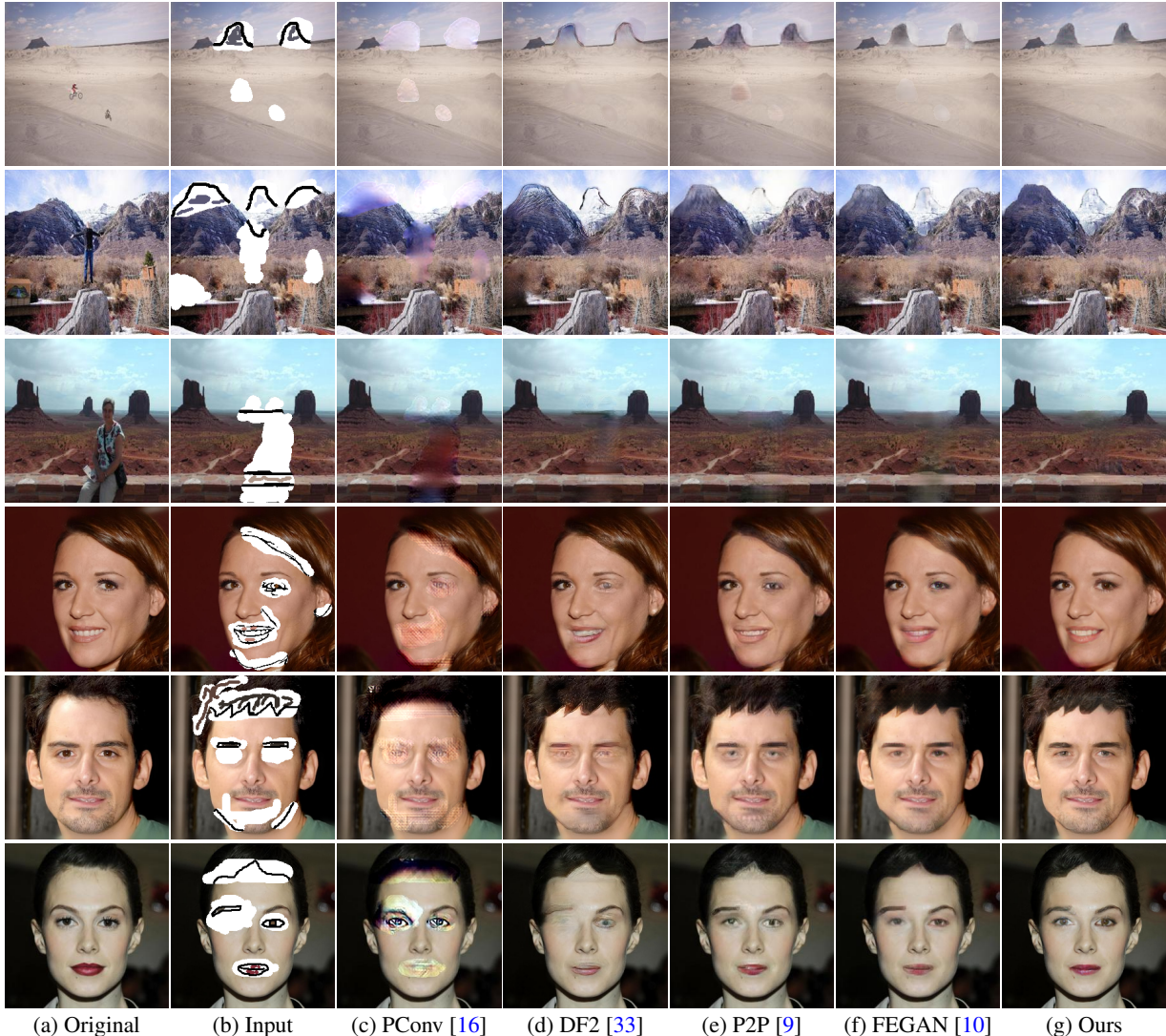
We evaluate existing editing methods including SC-FEGAN [10], Partial Conv [16], Deepfill2 [33] and Pix2Pix [9]. All these methods are retrained using the official implementations on the same datasets with the same input configurations for fair comparisons. The only difference between DeFLOCNet and other methods is that we send low-level controls into skip-connection layers rather than the encoder.

Visual Evaluations. Fig. 6 shows the visual comparison results. The original clean images are shown in (a). The input of existing methods is shown in (b). For a straightforward display, we combine input images with masks and low-level controls together. The results produced by Partial Conv and Deepfill2 are shown in (c) and (d) where structure distortions and blurry textures exist. This is because these two methods tend to incorporate neighboring content when filling the hole regions. They are not effective to generate meaningful content given by the users. In comparison, the results generated by Pix2Pix and SC-FEGAN are improved as shown in (e) and (f). These two methods focus more on the user controls and utilize adversarial learning to generate perceptual realistic contents. However, as these methods attach user controls to color images for the network input, structure generation is thus limited. The feature representations are not sufficient to convey both color images and low-level controls since their data distributions are extremely unbalanced. The structures of the eye regions shown in the last three rows are not effectively generated in (e) and (f).

Unlike existing methods that combine all the inputs together, DeFLOCNet sends the color image into the encoder and low-level controls into skip-connection layers via SGBs. These controls gradually enrich encoder features in each skip-connection layer, and refine these features from coarse to fine across multiple skip-connection layers. The results of our method are shown in (g) where image content is effectively generated with detailed textures.

Numerical Evaluations. We evaluate existing methods on the two benchmark datasets numerically from two aspects. First, we use standard metrics including PSNR, SSIM, and FID [7] to measure the pixel-level, structure-level, and holistic-level similarities between the output results and original images. When producing these results, we use the sketch lines and color dots from the hole regions without modifications. Table 1 shows the evaluation results where our method performs favorably against the existing methods. This indicates that our method is more effective to generate user-intended contents while maintaining visual pleasantness.

Besides standard metrics, we perform a human subject evaluation. There are over 20 volunteers to evaluate the re-



(a) Original (b) Input (c) PConv [16] (d) DF2 [33] (e) P2P [9] (f) FEGAN [10] (g) Ours

Figure 6. Visual comparisons with state-of-the-art methods. Original images are in (a). Input images are in (b) with low-level controls in the hole regions. Our method is effective to generate user-intended and visually pleasant contents in (g).

Table 1. Numerical evaluations on CelebA-HQ and Places2 datasets. ↓ indicates lower is better while ↑ indicates higher is better.

Model	CelebA-HQ					Places2				
	PConv [16]	DF2 [33]	SC-FEGAN [10]	P2P [9]	Ours	PConv [16]	DF2 [33]	SC-FEGAN [10]	P2P [9]	Ours
PSNR↑	22.12	24.48	24.56	25.17	25.42	20.14	22.0	23.57	21.90	24.30
SSIM↑	0.84	0.89	0.88	0.89	0.90	0.62	0.67	0.75	0.68	0.77
FID↓	24.34	23.88	14.63	12.45	9.92	201.09	113.54	77.82	93.46	63.56
Human↑	0.2%	0.8%	21.3%	15.1%	54.2%	0.0%	3.7%	18.2%	4.5%	73.6%

sults on both CelebA-HQ and Places2 datasets. The volunteers all have image processing background. There are 15 rounds for each subject. In each round, the subject needs to select the most visually pleasant result from the 5 results generated by existing methods without knowing the hole re-

gion in advance. We tally the votes and show the statistics in the last row in Table 1. The comparison results with respect to existing methods indicate that our method is more effective to generate visually high-quality image content.

Table 2. Ablation studies on CelebA-HQ and Places2. We show the numerical results produced by reducing injection times, reducing SGB amounts, removing sketch line guidance, removing texture generation branch, and our complete framework.

Model	CelebA-HQ					Places2				
	1 Inject	1 block	w/o 1- σ	w/o texture	Ours	1 Inject	1 block	w/o 1- σ	w/o texture	Ours
PSNR \uparrow	23.56	24.49	24.39	25.26	25.42	20.81	23.55	23.16	23.91	24.30
SSIM \uparrow	0.86	0.87	0.88	0.89	0.90	0.67	0.74	0.72	0.70	0.77
FID \downarrow	17.63	17.56	14.94	10.94	9.92	105.79	83.52	77.80	69.66	63.56



(a) Input (b) 1 inject (c) 1 SGB (d) w/o 1- σ (e) Ours

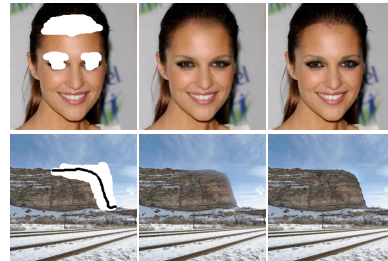
Figure 7. Ablation analysis on SGB. Input images are in (a). The result produced by using only 1 injection in each SGB is in (b), by using only one SGB is in (c), and by not using sketch line guidance (*i.e.*, $1 - \sigma$ in Eq. 4) is in (d). User intentions are not effectively reflected in these results in comparison to ours in (e).

4.2. Ablation Study

Our DeFLOCNet improves baseline results via SGBs and TGB. We analyze the effects of SGB and TGB on the output results.

Structure generation block. We set SGBs across multiple skip connection layers to reinforce user intentions from coarse to fine. Meanwhile, within each skip connection layers we gradually enrich encoder feature representations. Fig. 7 shows two visual examples. Input images are in (a), and we only use 1 injection within each SGB to produce the results in (b). On the other side, we only use one SGB set on the middle-level skip connection layer, and generate the results in (c). The results in (b) and (c) indicate that using limited injections or SGBs are not effective during structure generation. In contrast to these two configurations, we do not use sketch line constraint and produce the results in (d). They show that color propagates in arbitrary directions for unintended structure generation. By using more injections and propagation guidance, we are able to produce visually pleasant structures in (e). Table 2 numerically shows that multiple injections and propagation constraints improve the structure quality of the generated content.

Texture generation branch. We analyze the effect of TGB by comparing the results produced with TGB and without TGB. Fig. 8 shows two examples. Input images with user controls in (a). Without TGB, texture details are blurry in some regions in (b) (*e.g.*, the forelock hair and mountain



(a) Input (b) w/o TGB (c) w/ TGB

Figure 8. Ablation analysis on TGB. Inputs are in (a). The results of our method without using TGB are shown in (b). The results produced by using TGB are shown in (c) where there are more textures and details.

boundaries). The utilization of TGB synthesizes texture details based on the input image and thus reduces the blurry artifacts in (c). The numerical evaluation in Table 2 also indicates that TGB improves the generated content.

5. Concluding Remarks

We propose structure generation blocks set on skip connection layers to receive low-level controls, while only color images are sent to the encoder. The encoder features, representing only color images, are modulated via these blocks gradually to reinforce user intentions within each skip connection layer. Furthermore, the modulated encoder features with structure ingredients supplement the decoder features together with the generated texture features across multiple skip connection layers. Therefore, both structures and textures are generated from coarse to fine in the CNN feature space, bringing both user-intended and visually pleasing image content. The Experiments on the benchmark datasets indicate the effectiveness of our methods both numerically and visually compared to the state-of-the-art approaches.

6. Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under grant 62072169.

This work was partially supported by an ECS grant from the Research Grants Council of the Hong Kong (Project No. CityU 21209119) and an APRC grant from CityU, Hong Kong (Project No. 9610488).

References

- [1] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *ACM Transactions on Graphics*, 2019.
- [2] Jianbo Chen, Yelong Shen, Jianfeng Gao, Jingjing Liu, and Xiaodong Liu. Language-based image editing with recurrent attentive models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.
- [3] Haoye Dong, Xiaodan Liang, Yixuan Zhang, Xujie Zhang, Xiaohui Shen, Zhenyu Xie, Bowen Wu, and Jian Yin. Fashion editing with adversarial parsing learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [4] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Neural Information Processing Systems*, 2015.
- [5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, 2014.
- [6] Shuyang Gu, Jianmin Bao, Hao Yang, Dong Chen, Fang Wen, and Lu Yuan. Mask-guided portrait editing with conditional gans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.
- [7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems*, 2017.
- [8] Seunghoon Hong, Xinchun Yan, Thomas S Huang, and Honglak Lee. Learning hierarchical semantic image manipulation through structured representations. In *Neural Information Processing Systems*, 2018.
- [9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.
- [10] Youngjoo Jo and Jongyoul Park. Sc-fegan: Face editing generative adversarial network with user’s sketch and color. In *IEEE/CVF International Conference on Computer Vision*, 2019.
- [11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.
- [12] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. In *International Conference on Learning Representations*, 2018.
- [13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [14] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [15] Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang. Generative face completion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.
- [16] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *European Conference on Computer Vision*, 2018.
- [17] Hongyu Liu, Bin Jiang, Yibing Song, Wei Huang, and Chao Yang. Rethinking image inpainting via a mutual encoder-decoder with feature equalizations. In *European Conference on Computer Vision*, 2020.
- [18] Hongyu Liu, Bin Jiang, Yi Xiao, and Chao Yang. Coherent semantic attention for image inpainting. In *IEEE/CVF International Conference on Computer Vision*, 2019.
- [19] Seonghyeon Nam, Yunji Kim, and Seon Joo Kim. Text-adaptive generative adversarial networks: manipulating images with natural language. In *Neural Information Processing Systems*, 2018.
- [20] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.
- [21] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016.
- [22] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355*, 2016.
- [23] Tiziano Portenier, Qiyang Hu, Attila Szabo, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. Faceshop: Deep sketch-based face image editing. *arXiv preprint arXiv:1804.08972*, 2018.
- [24] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. *arXiv preprint arXiv:1907.10786*, 2019.
- [25] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson WH Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. In *IEEE/CVF International Conference on Computer Vision*, 2017.
- [26] Yibing Song, Chao Ma, Xiaohe Wu, Lijun Gong, Linchao Bao, Wangmeng Zuo, Chunhua Shen, Rynson WH Lau, and Ming-Hsuan Yang. Vital: Visual tracking via adversarial learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.
- [27] Yibing Song, Jiawei Zhang, Lijun Gong, Shengfeng He, Linchao Bao, Jinshan Pan, Qingxiang Yang, and Ming-Hsuan Yang. Joint face hallucination and deblurring via structure generation and detail enhancement. *International Journal of Computer Vision*, 2019.
- [28] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.
- [29] Yinglong Wang, Yibing Song, Chao Ma, and Bing Zeng. Rethinking image deraining via rain streaks and vapors. In *European Conference on Computer Vision*, 2020.
- [30] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *IEEE/CVF International Conference on Computer Vision*, 2015.

- [31] Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. Structure extraction from texture via relative total variation. *ACM Transactions on Graphics*, 2012.
- [32] Zhaoyi Yan, Xiaoming Li, Mu Li, Wangmeng Zuo, and Shiguang Shan. Shift-net: Image inpainting via deep feature rearrangement. In *European Conference on Computer Vision*, 2018.
- [33] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *IEEE/CVF International Conference on Computer Vision*, 2019.
- [34] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.