# EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning

Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z. Qureshi, Mehran Ebrahimi
Faculty of Science, University of Ontario Institute of Technology, Canada
{kamyar.nazeri, eric.ng, tony.joseph, faisal.qureshi, mehran.ebrahimi}@uoit.ca

## Abstract

*Over the last few years, deep learning techniques have yielded significant improvements in image inpainting. However, many of these techniques fail to reconstruct reasonable structures as they are commonly over-smoothed and/or blurry. This paper develops a new approach for image inpainting that does a better job of reproducing filled regions exhibiting fine details. We propose a two-stage adversarial model EdgeConnect that comprises of an edge generator followed by an image completion network. The edge generator hallucinates edges of the missing region (both regular and irregular) of the image, and the image completion network fills in the missing regions using hallucinated edges as a priori. We evaluate our model end-to-end over the publicly available datasets CelebA, Places2, and Paris StreetView, and show that it outperforms current state-of-the-art techniques quantitatively and qualitatively.*

## 1. Introduction

Image inpainting, or image completion, involves filling in missing regions of an image. It is an important step in many image editing tasks. It can, for example, be used to fill in the holes left after removing unwanted objects from an image. Humans have an uncanny ability to zero in on visual inconsistencies. Consequently, the filled regions must be perceptually plausible. Among other things, the lack of fine structure in the filled region is a giveaway that something is amiss, especially when the rest of the image contain sharp details. The work presented in this paper is motivated by our observation that many existing image inpainting techniques generate over-smoothed and/or blurry regions, failing to reproduce fine details.

We divide image inpainting into a two-stage process (Figure 1): edge generation and image completion. Edge generation is solely focused on hallucinating edges in the missing regions. The image completion network uses the hallucinated edges and estimates RGB pixel intensities of the missing regions. Both stages follow an adversarial framework [18] to ensure that the hallucinated edges and the RGB pixel intensities are visually consistent. Both networks incorporate losses based on deep features to enforce perceptually realistic results.

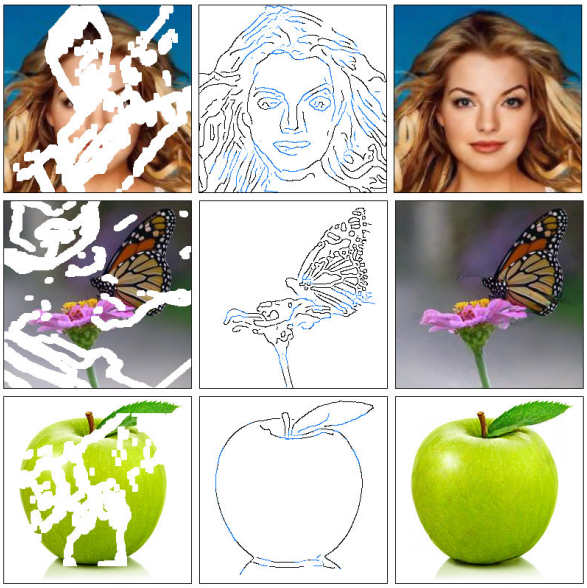


Figure 1: (Left) Input images with missing regions. The missing regions are depicted in white. (Center) Computed edge masks. Edges drawn in black are computed (for the available regions) using Canny edge detector; whereas edges shown in blue are hallucinated (for the missing regions) by the edge generator network. (Right) Image inpainting results of the proposed approach.

Like most computer vision problems, image inpainting predates the wide-spread use of deep learning techniques. Broadly speaking, traditional approaches for image inpainting can be divided into two groups: diffusion-based and patch-based. Diffusion-based methods propagate background data into the missing region by following a diffusive process typically modeled using differential operators [4, 14, 27, 2]. Patch-based methods, on the other hand, fill in missing regions with patches from a collection of source images that maximize patch similarity [7, 21]. These meth-

ods, however, do a poor job of reconstructing complex details that may be local to the missing region.

More recently deep learning approaches have found remarkable success at the task of image inpainting. These schemes fill the missing pixels using learned data distribution. They are able to generate coherent structures in the missing regions, a feat that was nearly impossible for traditional techniques. While these approaches are able to generate missing regions with meaningful structures, the generated regions are often blurry or suffer from artifacts, suggesting that these methods struggle to reconstruct high frequency information accurately.

Then, how does one force an image inpainting network to generate fine details? Since image structure is well-represented in its edge mask, we show that it is possible to generate superior results by conditioning an image inpainting network on edges in the missing regions. Clearly, we do not have access to edges in the missing regions. Rather, we train an edge generator that hallucinates edges in these areas. Our approach of "lines first, color next" is partly inspired by our understanding of how artists work [13]. "*In line drawing, the lines not only delineate and define spaces and shapes; they also play a vital role in the composition*", says Betty Edwards, highlights the importance of sketches from an artistic viewpoint [12]. Edge recovery, we suppose, is an easier task than image completion. Our proposed model essentially decouples the recovery of high and low-frequency information of the inpainted region.

We evaluate our proposed model on standard datasets CelebA [30], Places2 [56], and Paris StreetView [8]. We compare the performance of our model against current state-of-the-art schemes. Furthermore, we provide results of experiments carried out to study the effects of edge information on the image inpainting task. Our paper makes the following contributions:

- An edge generator capable of hallucinating edges in missing regions given edges and grayscale pixel intensities of the rest of the image.
- An image completion network that combines edges in the missing regions with color and texture information of the rest of the image to fill the missing regions.
- An end-to-end trainable network that combines edge generation and image completion to fill in missing regions exhibiting fine details.

We show that our model can be used in some common image editing applications, such as object removal and scene generation. Our source code is available at:
https://github.com/knazeri/edge-connect

## 2. Related Work

*Diffusion-based* methods propagate neighboring information into the missing regions [4, 2]. [14] adapted the Mumford-Shah segmentation model for image inpainting by introducing Euler's Elastica. However, reconstruction is restricted to locally available information for these diffusion-based methods, and these methods fail to recover meaningful structures in the missing regions. These methods also cannot adequately deal with large missing regions.

*Patch-based* methods fill in missing regions (*i.e.*, targets) by copying information from similar regions (*i.e.*, sources) of the same image (or a collection of images). Source regions are often blended into the target regions to minimize discontinuities [7, 21]. These methods are computationally expensive since similarity scores must be computed for every target-source pair. PatchMatch [3] addressed this issue by using a fast nearest neighbor field algorithm. These methods, however, assume that the texture of the inpainted region can be found elsewhere in the image. This assumption does not always hold. Consequently, these methods excel at recovering highly patterned regions such as background completion but struggle at reconstructing patterns that are locally unique.

One of the first *deep learning* methods designed for image inpainting is context encoder [38], which uses an encoder-decoder architecture. The encoder maps an image with missing regions to a low-dimensional feature space, which the decoder uses to construct the output image. However, the recovered regions of the output image often contain visual artifacts and exhibit blurriness due to the information bottleneck in the channel-wise fully connected layer. This was addressed by Iizuka *et al*. [22] by reducing the number of downsampling layers, and replacing the channel-wise fully connected layer with a series of dilated convolution layers [51]. The reduction of downsampling layers are compensated by using varying dilation factors. However, training time was increased significantly[1] due to extremely sparse filters created using large dilation factors. Yang *et al*. [49] uses a pre-trained VGG network [42] to improve the output of the context-encoder, by minimizing the feature difference of image background. This approach requires solving a multi-scale optimization problem iteratively, which noticeably increases computational cost during inference time. Liu *et al*. [28] introduced "partial convolution" for image inpainting, where convolution weights are normalized by the mask area of the window that the convolution filter currently resides over. This effectively prevents the convolution filters from capturing too many zeros when they traverse over the incomplete region.

Recently, several methods were introduced by providing additional information prior to inpainting. Yeh *et al*. [50] trains a GAN for image inpainting with uncorrupted data. During inference, back-propagation is employed for $1,500$ iterations to find the representation of the corrupted image on a uniform noise distribution. However, the model is slow during inference since back-propagation must be performed

[1] Model by [22] required two months of training over four GPUs.

for every image it attempts to recover. Dolhansky and Ferrer [9] demonstrate the importance of exemplar information for inpainting. Their method is able to achieve both sharp and realistic inpainting results. Their method, however, is geared towards filling in missing eye regions in frontal human face images. It is highly specialized and does not generalize well. Contextual Attention [53] takes a two-step approach to the problem of image inpainting. First, it produces a coarse estimate of the missing region. Next, a refinement network sharpens the result using an attention mechanism by searching for a collection of background patches with the highest similarity to the coarse estimate. [43] takes a similar approach and introduces a "patch-swap" layer which replaces each patch inside the missing region with the most similar patch on the boundary. These schemes suffer from two limitations: 1) the refinement network assumes that the coarse estimate is reasonably accurate, and 2) these methods cannot handle missing regions with arbitrary shapes. Free-form inpainting method proposed in [52] is perhaps closest in spirit to our scheme. It uses hand-drawn sketches to guide the inpainting process. Our method does away with hand-drawn sketches and instead learns to hallucinate edges in the missing regions.

### 2.1. Image-to-Edges vs. Edges-to-Image

The inpainting technique proposed in this paper subsumes two disparate computer vision problems: Image-to-Edges and Edges-to-Image. There is a large body of literature that addresses "Image-to-Edges" problems [5, 10, 26, 29]. Canny edge detector, an early scheme for constructing edge maps, for example, is roughly 30 years old [6]. Dollár and Zitnikc [11] use *structured learning* [35] on random decision forests to predict local edge masks. Holistically-nested Edge Detection (HED) [48] is a fully convolutional network that learns edge information based on its importance as a feature of the overall image. In our work, we train on edge maps computed using Canny edge detector. We explain this in detail in Section 4.1 and Section 5.3.

Traditional "Edges-to-Image" methods typically follow a bag-of-words approach, where image content is constructed through a pre-defined set of keywords. These methods, however, are unable to accurately construct fine-grained details especially near object boundaries. Scribbler [41] is a learning-based model where images are generated using line sketches as the input. The results of their work possess an art-like quality, where color distribution of the generated result is guided by the use of color in the input sketch. Isola *et al*. [23] proposed a conditional GAN framework [33], called pix2pix, for image-to-image translation problems. This scheme can use available edge information as *a priori*. CycleGAN [57] extends this framework and finds a reverse mapping back to the original data distribution. This approach yields superior results since the aim is to learn the inverse of the forward mapping.

## 3. EdgeConnect

We propose an image inpainting network that consists of two stages: 1) edge generator, and 2) image completion network (Figure 2). Both stages follow an adversarial model [18], *i.e*. each stage consists of a generator/discriminator pair. Let $G_1$ and $D_1$ be the generator and discriminator for the edge generator, and $G_2$ and $D_2$ be the generator and discriminator for the image completion network, respectively. To simplify notation, we will use these symbols also to represent the function mappings of their respective networks.

Our generators follow an architecture similar to the method proposed by Johnson *et al*. [24], which has achieved impressive results for style transfer, super-resolution [40, 17], and image-to-image translation [57]. Specifically, the generators consist of encoders that down-sample twice, followed by eight residual blocks [19] and decoders that up-sample images back to the original size. Dilated convolutions with a dilation factor of two are used instead of regular convolutions in the residual layers, resulting in a receptive field of 205 at the final residual block. For discriminators, we use a $70 \times 70$ PatchGAN [23, 57] architecture, which determines whether or not overlapping image patches of size $70 \times 70$ are real. We use instance normalization [45] across all layers of the network[2].

### 3.1. Edge Generator

Let $\mathbf{I}_{gt}$ be ground truth images. Their edge map and grayscale counterpart will be denoted by $\mathbf{C}_{gt}$ and $\mathbf{I}_{gray}$, respectively. In the edge generator, we use the masked grayscale image $\tilde{\mathbf{I}}_{gray} = \mathbf{I}_{gray} \odot (\mathbf{1} - \mathbf{M})$ as the input, its edge map $\tilde{\mathbf{C}}_{gt} = \mathbf{C}_{gt} \odot (\mathbf{1} - \mathbf{M})$, and image mask $\mathbf{M}$ as a pre-condition (1 for the missing region, 0 for background). Here, $\odot$ denotes the Hadamard product. The generator predicts the edge map for the masked region

$$\mathbf{C}_{pred} = G_1\left(\tilde{\mathbf{I}}_{gray}, \tilde{\mathbf{C}}_{gt}, \mathbf{M}\right). \tag{1}$$

We use $\mathbf{C}_{gt}$ and $\mathbf{C}_{pred}$ conditioned on $\mathbf{I}_{gray}$ as inputs of the discriminator that predicts whether or not an edge map is real. The network is trained with an objective comprised of an adversarial loss and feature-matching loss [46]

$$\min_{G_1}\max_{D_1}\mathcal{L}_{G_1} = \min_{G_1}\left(\lambda_{adv,1}\max_{D_1}\left(\mathcal{L}_{adv,1}\right) + \lambda_{FM}\mathcal{L}_{FM}\right) \tag{2}$$

where $\lambda_{adv,1}$ and $\lambda_{FM}$ are regularization parameters. The adversarial loss is defined as

$$\mathcal{L}_{adv,1} = \mathbb{E}_{(\mathbf{C}_{gt},\mathbf{I}_{gray})}\left[\log D_1(\mathbf{C}_{gt},\mathbf{I}_{gray})\right] + \mathbb{E}_{\mathbf{I}_{gray}}\log\left[1 - D_1(\mathbf{C}_{pred},\mathbf{I}_{gray})\right]. \tag{3}$$

[2]The details of our architecture are in appendix A

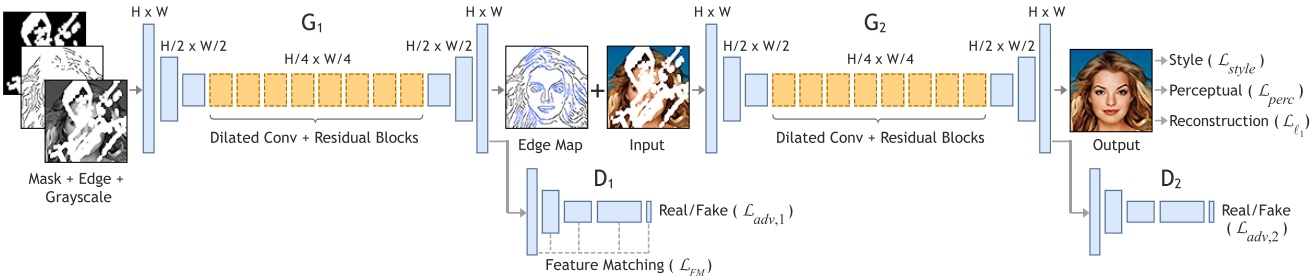



Figure 2: Summary of our proposed method. Incomplete grayscale image and edge map, and mask are the inputs of $G_1$ to predict the full edge map. Predicted edge map and incomplete color image are passed to $G_2$ to perform the inpainting task.

The feature-matching loss $\mathcal{L}_{FM}$ compares the activation maps in the intermediate layers of the discriminator. This stabilizes the training process by forcing the generator to produce results with representations that are similar to real images. This is similar to perceptual loss [24, 16, 15], where activation maps are compared with those from the pre-trained VGG network. However, since the VGG network is not trained to produce edge information, it fails to capture the result that we seek in the initial stage. The feature matching loss $\mathcal{L}_{FM}$ is defined as

$$\mathcal{L}_{FM} = \mathbb{E}\left[\sum_{i=1}^{L} \frac{1}{N_i} \left\| D_1^{(i)}(\mathbf{C}_{gt}) - D_1^{(i)}(\mathbf{C}_{pred}) \right\|_1\right], \quad (4)$$

where $L$ is the final convolution layer of the discriminator, $N_i$ is the number of elements in the $i$'th activation layer, and $D_1^{(i)}$ is the activation in the $i$'th layer of the discriminator. Spectral normalization (SN) [34] further stabilizes training by scaling down weight matrices by their respective largest singular values, effectively restricting the Lipschitz constant of the network to one. Although this was originally proposed to be used only on the discriminator, recent works [54, 36] suggest that generator can also benefit from SN by suppressing sudden changes of parameter and gradient values. Therefore, we apply SN to both generator and discriminator. Spectral normalization was chosen over Wasserstein GAN (WGAN), [1] as we found that WGAN was several times slower in our early tests. Note that only $\mathcal{L}_{adv,1}$ is maximized over $D_1$ since $D_1$ is used to retrieve activation maps for $\mathcal{L}_{FM}$. For our experiments, we choose $\lambda_{adv,1} = 1$ and $\lambda_{FM} = 10$.

### 3.2. Image Completion Network

The image completion network uses the incomplete color image $\tilde{\mathbf{I}}_{gt} = \mathbf{I}_{gt} \odot (\mathbf{1} - \mathbf{M})$ as input, conditioned using a composite edge map $\mathbf{C}_{comp}$. The composite edge map is constructed by combining the background region of ground truth edges with generated edges in the corrupted region from the previous stage, *i.e.* $\mathbf{C}_{comp} = \mathbf{C}_{gt} \odot (\mathbf{1} - \mathbf{M}) + \mathbf{C}_{pred} \odot \mathbf{M}$. The network returns a color image $\mathbf{I}_{pred}$, with missing regions filled in, that has the same resolution as the input image:

$$\mathbf{I}_{pred} = G_2\left(\tilde{\mathbf{I}}_{gt}, \mathbf{C}_{comp}\right). \quad (5)$$

This is trained over a joint loss that consists of an $\ell_1$ loss, adversarial loss, perceptual loss, and style loss. To ensure proper scaling, the $\ell_1$ loss is normalized by the mask size. The adversarial loss is defined similar to Eq. 3, as

$$\mathcal{L}_{adv,2} = \mathbb{E}_{(\mathbf{I}_{gt}, \mathbf{C}_{comp})}\left[\log D_2(\mathbf{I}_{gt}, \mathbf{C}_{comp})\right] + \mathbb{E}_{\mathbf{C}_{comp}} \log\left[1 - D_2(\mathbf{I}_{pred}, \mathbf{C}_{comp})\right]. \quad (6)$$

We include the two losses proposed in [16, 24] commonly known as perceptual loss $\mathcal{L}_{perc}$ and style loss $\mathcal{L}_{style}$. As the name suggests, $\mathcal{L}_{perc}$ penalizes results that are not perceptually similar to labels by defining a distance measure between activation maps of a pre-trained network. Perceptual loss is defined as

$$\mathcal{L}_{perc} = \mathbb{E}\left[\sum_i \frac{1}{N_i} \left\|\phi_i(\mathbf{I}_{gt}) - \phi_i(\mathbf{I}_{pred})\right\|_1\right] \quad (7)$$

where $\phi_i$ is the activation map of the $i$'th layer of a pre-trained network. For our work, $\phi_i$ corresponds to activation maps from layers `relu1_1`, `relu2_1`, `relu3_1`, `relu4_1` and `relu5_1` of the VGG-19 network pre-trained on the ImageNet dataset [39]. These activation maps are also used to compute style loss which measures the differences between covariances of the activation maps. Given feature maps of sizes $C_j \times H_j \times W_j$, style loss is computed by

$$\mathcal{L}_{style} = \mathbb{E}_j\left[\left\|G_j^{\phi}(\tilde{\mathbf{I}}_{pred}) - G_j^{\phi}(\tilde{\mathbf{I}}_{gt})\right\|_1\right] \quad (8)$$

where $G_j^{\phi}$ is a $C_j \times C_j$ Gram matrix constructed from activation maps $\phi_j$. We choose to use style loss as it was shown by Sajjadi *et al.* [40] to be an effective tool to combat "checkerboard" artifacts caused by transpose convolution layers [37]. Our overall loss function is

$$\mathcal{L}_{G_2} = \lambda_{\ell_1}\mathcal{L}_{\ell_1} + \lambda_{adv,2}\mathcal{L}_{adv,2} + \lambda_p\mathcal{L}_{perc} + \lambda_s\mathcal{L}_{style}. \quad (9)$$

(a) Ground Truth (b) Masked Image (c) Yu *et al*. [53] (d) Iizuka *et al*. [22] (e) Ours (f) Ours (Canny)

Figure 3: Comparison of qualitative results with existing models. (a) Ground Truth Image. (b) Ground Truth with Mask. (c) Yu *et al*. [53]. (d) Iizuka *et al*. [22]. (e) Ours (end-to-end). (f) Ours ($G_2$ only with Canny $\sigma = 2$).

For our experiments, we choose $\lambda_{\ell_1} = 1$, $\lambda_{adv,2} = \lambda_p = 0.1$, and $\lambda_s = 250$. We noticed that the training time increases significantly if spectral normalization is included. We believe this is due to the network becoming too restrictive with the increased number of terms in the loss function. Therefore we choose to exclude spectral normalization from the image completion network.

# 4. Experiments

## 4.1. Edge Information and Image Masks

To train $G_1$, we generate training labels (*i.e*. edge maps) using Canny edge detector. The sensitivity of Canny edge detector is controlled by the standard deviation of the Gaussian smoothing filter $\sigma$. For our tests, we empirically found that $\sigma \approx 2$ yields the best results (Figure 6). In Section 5.3, we investigate the effect of the quality of edge maps on the overall image completion.

For our experiments, we use two types of image masks: regular and irregular. Regular masks are square masks of fixed size (25% of total image pixels) centered at a random location within the image. We obtain irregular masks from the work of Liu *et al*. [28]. Irregular masks are augmented by introducing four rotations $(0°, 90°, 180°, 270°)$ and a horizontal reflection for each mask. They are classified based on their sizes relative to the entire image in increments of $10\%$ (*e.g*., 0-10%, 10-20%, *etc*.).

## 4.2. Training Setup and Strategy

Our proposed model is implemented in PyTorch. The network is trained using $256 \times 256$ images with a batch size of eight. The model is optimized using Adam optimizer [25] with $\beta_1 = 0$ and $\beta_2 = 0.9$. Generators $G_1, G_2$ are trained separately using Canny edges with learning rate $10^{-4}$ until the losses plateau. We lower the learning rate to $10^{-5}$ and continue to train $G_1$ and $G_2$ until convergence. Finally, we fine-tune the networks by removing $D_1$, then train $G_1$ and $G_2$ end-to-end with learning rate $10^{-6}$ until convergence. Discriminators are trained with a learning rate one tenth of the generators'.

# 5. Results

Our proposed model is evaluated on the datasets CelebA [30], Places2 [56], and Paris StreetView [8]. Results are compared against the current state-of-the-art methods both qualitatively and quantitatively.

## 5.1. Qualitative Comparison

Figure 4 shows a sample of images generated by our model. For visualization purposes, we reverse the colors of $\mathbf{C}_{comp}$. Our model is able to generate photo-realistic results with a large fraction of image structures remaining intact. Furthermore, by including style loss, the inpainted images lack any "checkerboard" artifacts in the generated results. As importantly, the inpainted images exhibit minimal blurriness. Figure 3 compares images generated by our method with those generated by other state-of-the-art techniques. The images generated by our proposed model are closer to ground truth than images from other methods. We conjecture that when edge information is present, the network only needs to learn the color distribution, without having to worry about preserving image structure.

## 5.2. Quantitative Comparison

**Numerical Metrics** We measure the quality of our results using the following metrics: 1) relative $\ell_1$; 2) structural similarity index (SSIM) [47], with a window size of 11; and 3) peak signal-to-noise ratio (PSNR). These metrics assume pixel-wise independence, which may assign favorable scores to perceptually inaccurate results. Therefore, we also include Fréchet Inception Distance (FID) [20]. Re-

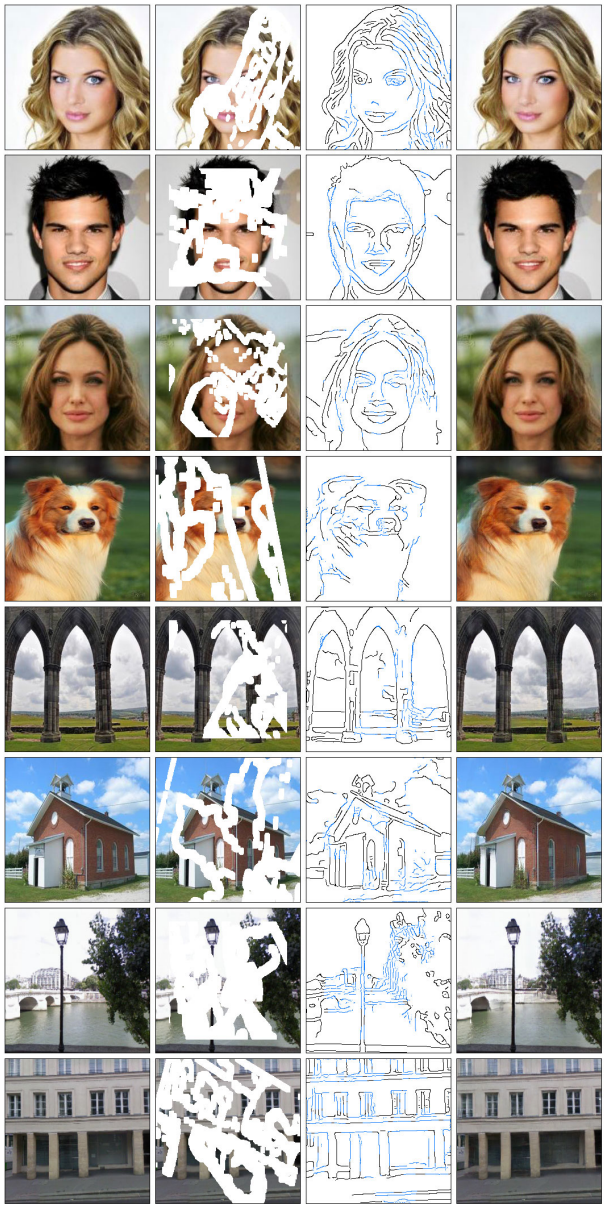


Figure 4: (Left to Right) Original image, input image, generated edges, inpainted results without any post-processing.

| | Mask | CA | GLCIC | PConv* | Ours | Canny |
|---|---|---|---|---|---|---|
| $\ell_1$ (%)† | 10-20% | 2.41 | 2.66 | **1.14** | 1.50 | 1.16 |
| | 20-30% | 4.23 | 4.70 | **1.98** | 2.59 | 1.88 |
| | 30-40% | 6.15 | 6.78 | **3.02** | 3.77 | 2.60 |
| | 40-50% | 8.03 | 8.85 | **4.11** | 5.14 | 3.41 |
| | Fixed | 4.37 | 4.12 | - | **3.86** | 2.22 |
| SSIM⋆ | 10-20% | 0.893 | 0.862 | 0.869 | **0.920** | 0.941 |
| | 20-30% | 0.815 | 0.771 | 0.777 | **0.861** | 0.902 |
| | 30-40% | 0.739 | 0.686 | 0.685 | **0.799** | 0.863 |
| | 40-50% | 0.662 | 0.603 | 0.589 | **0.731** | 0.821 |
| | Fixed | 0.818 | 0.814 | - | **0.823** | 0.892 |
| PSNR⋆ | 10-20% | 24.36 | 23.49 | **28.02** | 27.95 | 30.85 |
| | 20-30% | 21.19 | 20.45 | 24.90 | **24.92** | 28.35 |
| | 30-40% | 19.13 | 18.50 | 22.45 | **22.84** | 26.66 |
| | 40-50% | 17.75 | 17.17 | 20.86 | **21.16** | 25.20 |
| | Fixed | 20.65 | 21.34 | - | **21.75** | 26.52 |
| FID† | 10-20% | 6.16 | 11.84 | - | **2.32** | 2.25 |
| | 20-30% | 14.17 | 25.11 | - | **4.91** | 3.42 |
| | 30-40% | 24.16 | 39.88 | - | **8.91** | 4.87 |
| | 40-50% | 35.78 | 54.30 | - | **14.98** | 7.13 |
| | Fixed | 8.31 | 8.42 | - | **8.16** | 3.24 |

Table 1: Quantitative results over Places2 with models: Contextual Attention (CA) [53], Globally and Locally Consistent Image Completion (GLCIC) [22], Partial Convolution (PConv) [28], $G_1$ and $G_2$ (Ours), $G_2$ only with Canny edges (Canny). The best result of each row is boldfaced except for Canny. *Values taken from the paper [28]. †Lower is better. ⋆Higher is better.

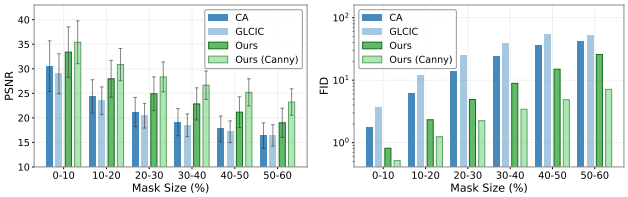



Figure 5: Effect of mask sizes on PSNR and FID. (Places2)

cent works [55, 54, 9] have shown that metrics based on deep features are closer to those based on human perception. FID measures the Wasserstein-2 distance between the feature space representations of real and inpainted images using a pre-trained Inception-V3 model [44]. The results over Places2 dataset are reported in Table 1. Note that these statistics are based on the synthesized image which mostly comprises of the ground truth image. Therefore our reported FID values are lower than other generative models reported in [31].

Figure 5 shows the performance of our model for various mask sizes. Statistics for competing techniques are obtained using their respective pre-trained weights, where available[3] [4]. Results for Partial Convolution (PConv) [28] are taken from their paper as the source code is not available at the time of writing. Note that $\ell_1$ (%) errors for PConv are lower than those achieved by our method and those reported in CA [53] and GLCIC [22]. While we are not sure why this is so, we suspect PConv is computing this score differently than how we compute it. Our statistics are calculated over $10,000$ random images in the test set.

[3] https://github.com/JiahuiYu/generative_inpainting

[4] https://github.com/satoshiiizuka/siggraph2017_inpainting

**Visual Turing Tests** We evaluate our results using the human perceptual metrics *2 alternative forced choice* (2AFC) and *just noticeable differences* (JND). For 2AFC, we ask participants whether or not an image is real from a pool of randomized images. For JND, we ask participants to select the more realistic image from pairs of real and generated images. Participants are given two seconds for each test. The tests are performed over 300 images for each model and mask size. Each image is shown 10 times in total. The results are summarized in Table 2.

| | Mask | CA | GLCIC | Ours |
|---|---|---|---|---|
| JND (%) | 10-20% | 21 ± 1.2% | 16.9 ± 1.1% | **39.7 ± 1.5%** |
| | 20-30% | 15.5 ± 1.1% | 14.3 ± 1% | **37 ± 1.5%** |
| | 30-40% | 12.9 ± 1% | 12.3 ± 1% | **27.5 ± 1.3%** |
| | 40-50% | 12.7 ± 2% | 10.9 ± 0.9% | **25.4 ± 1.3%** |
| 2AFC (%) | 10-20% | 38.7 ± 1.8% | 22.5 ± 1.5% | **88.7 ± 1.2%** |
| | 20-30% | 23.4 ± 1.5% | 12.1 ± 1.2% | **77.6 ± 1.5%** |
| | 30-40% | 13.5 ± 1.3% | 4.3 ± 0.7% | **66.4 ± 1.8%** |
| | 40-50% | 9.9 ± 1% | 2.8 ± 0.6% | **58 ± 1.8%** |

Table 2: 2AFC and JND scores for various mask sizes on Places2. 2AFC score for ground truth is **94.6 ± 0.5%**.

### 5.3. Ablation Study

**Quantity of Edges versus Inpainting Quality** We now turn our attention to the key assumption of this work: edge information helps with image inpainting. Table 3 shows inpainting results with and without edge information. Our model achieved better scores for every metric when edge information was incorporated into the inpainting model, even when a significant portion of the image is missing.

| | CelebA | | Places2 | |
|---|---|---|---|---|
| Edges | No | Yes | No | Yes |
| $\ell_1$ (%) | 4.11 | 3.03 | 6.69 | 5.14 |
| SSIM | 0.802 | 0.846 | 0.682 | 0.731 |
| PSNR | 23.33 | 25.28 | 19.59 | 21.16 |
| FID | 6.16 | 2.82 | 32.18 | 14.98 |

Table 3: Comparison of inpainting results with edge information (our full model) and without edge information ($G_2$ only, trained without edges). Statistics are based on $10,000$ random masks with size 40-50% of the entire image.

Next, we turn to a more interesting question: How much edge information is needed to see improvements in the generated images? We again use Canny edge detector to construct edge information. We use the parameter $\sigma$ to control the amount of edge information available to the image completion network. Specifically, we train our image completion network using edge maps generated for $\sigma = 0, 0.5, \ldots, 5.5$, and we found that the best image inpainting results are obtained with edges corresponding to $\sigma \in [1.5, 2.5]$, across all datasets shown in Figure 6. For large values of $\sigma$, too few edges are available to make a difference in the quality of generated images. On the other hand, when $\sigma$ is too small, too many edges are produced, which adversely affect the quality of the generated images. We used this study to set $\sigma = 2$ when creating ground truth edge maps for the training of the edge generator network.

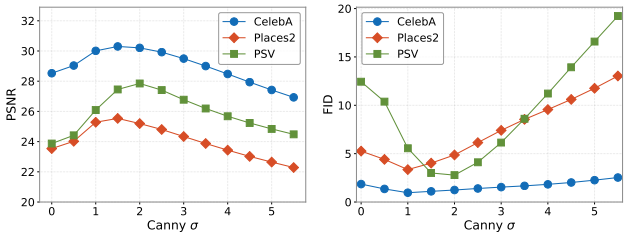



Figure 6: Effect of $\sigma$ in Canny detector on PSNR and FID.

Figure 7 shows how different values of $\sigma$ affects the inpainting task. Note that in a region where edge data is sparse, the quality of the inpainted region degrades. For instance, in the generated image for $\sigma = 5$, the left eye was reconstructed much sharper than the right eye.

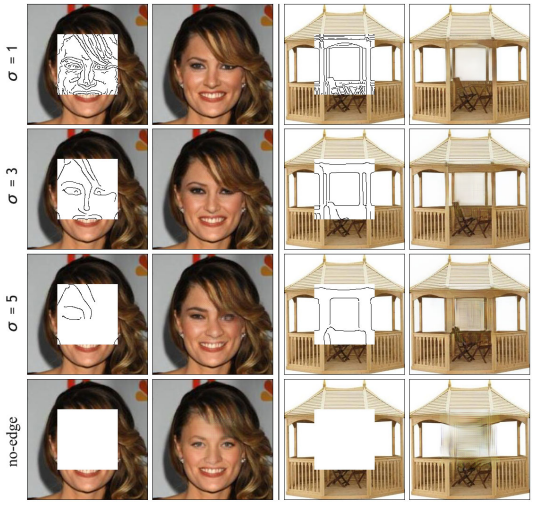



Figure 7: Effect of $\sigma$ in Canny edge detector on inpainting results. Top to bottom: $\sigma = 1, 3, 5$, no edge data.

**Alternative Edge Detection Systems** We use Canny edge detector to produce training labels for the edge generator network due to its speed, robustness, and ease of use. Canny edges are one-pixel wide, and are represented as binary masks (1 for edge, 0 for background). Edges produced with HED [48], however, are of varying thickness, and pix-

els can have intensities ranging between 0 and 1. We noticed that it is possible to create edge maps that look eerily similar to human sketches by performing element-wise multiplication on Canny and HED edge maps (Figure 8). We trained our image completion network using the combined edge map. However, we did not notice any improvements in the inpainting results.[5]

(a) (b) (c) (d)

Figure 8: (a) Image. (b) Canny. (c) HED. (d) Canny⊙HED.

## 6. Discussions and Future Work

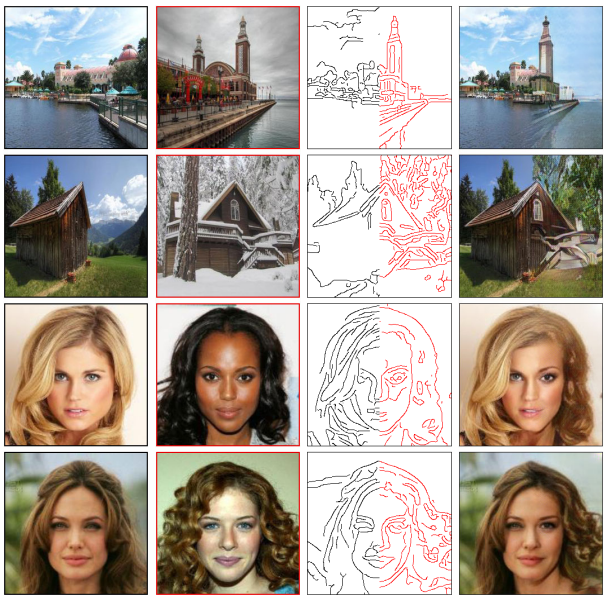
We proposed EdgeConnect, a new deep learning model for image inpainting tasks. EdgeConnect comprises of an edge generator and an image completion network, both following an adversarial model. We demonstrate that edge information plays an important role in the task of image inpainting. Our method achieves state-of-the-art results on standard benchmarks, and is able to deal with images with multiple, irregularly shaped missing regions.

The trained model can be used as an interactive image editing tool. We can, for example, manipulate objects in the edge domain and transform the edge maps back to generate a new image. This is demonstrated in Figure 10. Here we have removed the right-half of a given image to be used as input. The edge maps, however, are provided by a different image. The generated image seems to share characteristics of the two images. Figure 11 shows examples where we attempt to remove unwanted objects from existing images.

We plan to investigate better edge detectors. While effectively delineating the edges is more useful than hundreds of detailed lines, our edge generating model sometimes fails to accurately depict the edges in highly textured areas, or when a large portion of the image is missing (as seen in Figure 9). We believe our fully convolutional generative model can be extended to very high-resolution inpainting applications with an improved edge generating system.

Figure 9: Inpainted results where edge generator fails to produce relevant edge information.

[5]Further analysis with HED are available in appendix C.



(a) (b) (c) (d)

Figure 10: Edge-map (c) generated using the left-half of (a) (shown in black) and right-half of (b) (shown in red). Input is (a) with the right-half removed, producing the output (d).

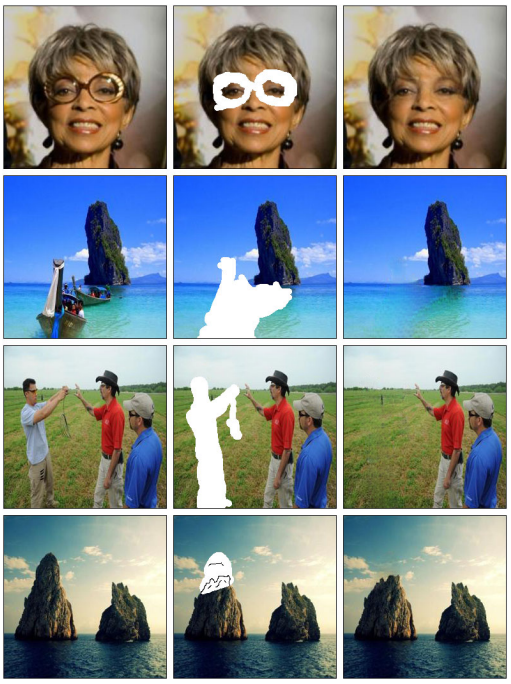


Figure 11: Examples of object removal and image editing using our EdgeConnect model. (Left) Original image. (Center) Unwanted object removed with optional edge information to guide inpainting. (Right) Generated image.

**Acknowledgments:** This research was supported in part by NSERC Discovery Grant. The authors would like to thank Konstantinos G. Derpanis for helpful discussions and feedbacks. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

## References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 4

[2] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE transactions on image processing*, 10(8):1200–1211, 2001. 1, 2

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on graphics (TOG)*, 28(3):24, 2009. 2

[4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, 2000. 1, 2

[5] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4380–4389, 2015. 3

[6] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 3

[7] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on graphics (TOG)*, 31(4):82–1, 2012. 1, 2

[8] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros. What makes paris look like paris? *ACM Transactions on graphics (TOG)*, 31(4), 2012. 2, 5

[9] B. Dolhansky and C. C. Ferrer. Eye in-painting with exemplar generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7902–7911, 2018. 3, 6

[10] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1964–1971. IEEE, 2006. 3

[11] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1558–1570, 2015. 3

[12] B. Edwards. *Drawing on the Right Side of the Brain: The Definitive, 4th Edition*. Penguin Publishing Group, 2012. 2

[13] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Transactions on graphics (TOG)*, 31(4):44–1, 2012. 2

[14] S. Esedoglu and J. Shen. Digital inpainting based on the mumford–shah–euler image model. *European Journal of Applied Mathematics*, 13(4):353–370, 2002. 1, 2

[15] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015. 4

[16] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. 4

[17] M. W. Gondal, B. Schölkopf, and M. Hirsch. The unreasonable effectiveness of texture transfer for single image super-resolution. In *Workshop and Challenge on Perceptual Image Restoration and Manipulation (PIRM) at the 15th European Conference on Computer Vision (ECCV)*, 2018. 3

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 1, 3

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 3

[20] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017. 5

[21] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Image completion using planar structure guidance. *ACM Transactions on graphics (TOG)*, 33(4):129, 2014. 1, 2

[22] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017. 2, 5, 6, 13

[23] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3, 11

[24] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 694–711. Springer, 2016. 3, 4, 11

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[26] Y. Li, M. Paluri, J. M. Rehg, and P. Dollár. Unsupervised learning of edges. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1619–1627, 2016. 3

[27] D. Liu, X. Sun, F. Wu, S. Li, and Y.-Q. Zhang. Image compression with edge-based inpainting. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(10):1273–1287, 2007. 1

[28] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *European Conference on Computer Vision (ECCV)*, September 2018. 2, 5, 6

[29] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai. Richer convolutional features for edge detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5872–5881. IEEE, 2017. 3

[30] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 2, 5

[31] M. Lui, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are gans created equal? a large-scale study. In *Advances in Neural Information Processing Systems*. 6

[32] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013. 11

[33] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 3

[34] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 4

[35] S. Nowozin, C. H. Lampert, et al. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011. 3

[36] A. Odena, J. Buckman, C. Olsson, T. B. Brown, C. Olah, C. Raffel, and I. Goodfellow. Is generator conditioning causally related to gan performance? In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 4

[37] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 1(10):e3, 2016. 4

[38] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2536–2544, 2016. 2

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 4

[40] M. S. M. Sajjadi, B. Scholkopf, and M. Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 4501–4510. IEEE, 2017. 3, 4

[41] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017. 3

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2

[43] Y. Song, C. Yang, Z. Lin, X. Liu, Q. Huang, H. Li, and C. Jay. Contextual-based image inpainting: Infer, match, and translate. In *European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 3

[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 6

[45] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[46] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 5, 2018. 3

[47] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 5

[48] S. Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1395–1403, 2015. 3, 7, 12

[49] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[50] R. A. Yeh, C. Chen, T.-Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[51] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016. 2

[52] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. *arXiv preprint arXiv:1806.03589*, 2018. 3

[53] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 5, 6, 13

[54] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 4, 6

[55] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[56] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 2, 5

[57] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 3, 11

# A. Network Architectures

## A.1. Generators

We follow a similar naming convention as those presented in [57]. Let `ck` denote a $7 \times 7$ Convolution-SpectralNorm-InstanceNorm-ReLU layer with $k$ filters and stride $1$ with reflection padding. Let `dk` denote a $4 \times 4$ Convolution-SpectralNorm-InstanceNorm-ReLU layer with $k$ filters and stride 2 for down-sampling. Let `uk` be defined in the same manner as `dk` with transpose convolution for up-sampling. Let `Rk` denote a residual block of channel size $k$ across both layers. We use dilated convolution in the first layer of `Rk` with dilation factor of 2, followed by spectral normalization and instance normalization.

The architecture of our generators is adopted from the model proposed by Johnson *et al.* [24]:
`c64, d128, d256, R256, R256, R256, R256, R256, R256, R256, R256, u128, u64, c*`.

The final layer `c*` varies depending on the generator. In the edge generator $G_1$, `c*` has channel size of 1 with sigmoid activation for edge prediction. In the image completion network $G_2$, `c*` has channel size of 3 with `tanh` (scaled) activation for the prediction of RGB pixel intensities. In addition, we remove spectral normalization from all layers of $G_2$.

## A.2. Discriminators

The discriminators $D_1$ and $D_2$ follow the same architecture based on the $70 \times 70$ PatchGAN [23, 57]. Let `Ck-s` denote a $4 \times 4$ Convolution-SpectralNorm-LeakyReLU layer with $k$ filters of stride $s$. The discriminators have the architecture `C64-2, C128-2, C256-2, C512-1, C1-1`. The final convolution layer produces scores predicting whether $70 \times 70$ overlapping image patches are real or fake. LeakyReLU [32] is employed with slope $0.2$.

# B. Experimental Results

We provide additional results produced by our model over the following datasets:

- CelebA ($202,599$ images)
- Places2 ($10$ million+ images)
- Paris StreetView ($14,900$ images)

With CelebA, we cropped the center $178 \times 178$ of the images, then resized them to $256 \times 256$ using bilinear interpolation. For Paris StreetView, since the images in the dataset are elongated ($936 \times 537$), we separate each image into three: 1) Left $537 \times 537$, 2) middle $537 \times 537$, 3) right $537 \times 537$, of the image. These images are scaled down to $256 \times 256$ for our model, totaling $44,700$ images.

## B.1. Accuracy of Edge Generator

Table 4 shows the accuracy of our edge generator $G_1$ across all three datasets. We measure precision and recall for various mask sizes. We emphasize that the goal of this experiment is not to achieve best precision and recall results, but instead to showcase how close the generated edges are to the oracle.

| | Mask | Precision | Recall |
|---|---|---|---|
| CelebA | 0-10% | 40.24 | 38.23 |
| | 10-20% | 34.28 | 34.05 |
| | 20-30% | 29.23 | 30.07 |
| | 30-40% | 24.92 | 25.88 |
| | 40-50% | 21.73 | 22.55 |
| | 50-60% | 16.39 | 16.93 |
| Places2 | 0-10% | 37.87 | 36.41 |
| | 10-20% | 32.66 | 32.53 |
| | 20-30% | 28.36 | 28.92 |
| | 30-40% | 25.02 | 25.38 |
| | 40-50% | 22.48 | 22.48 |
| | 50-60% | 18.14 | 17.76 |
| PSV | 0-10% | 49.46 | 46.76 |
| | 10-20% | 44.31 | 42.31 |
| | 20-30% | 39.38 | 37.80 |
| | 30-40% | 35.11 | 33.54 |
| | 40-50% | 31.29 | 29.60 |
| | 50-60% | 24.61 | 22.59 |

Table 4: Quantitative performance of our edge generator $G_1$ trained on Canny edges.

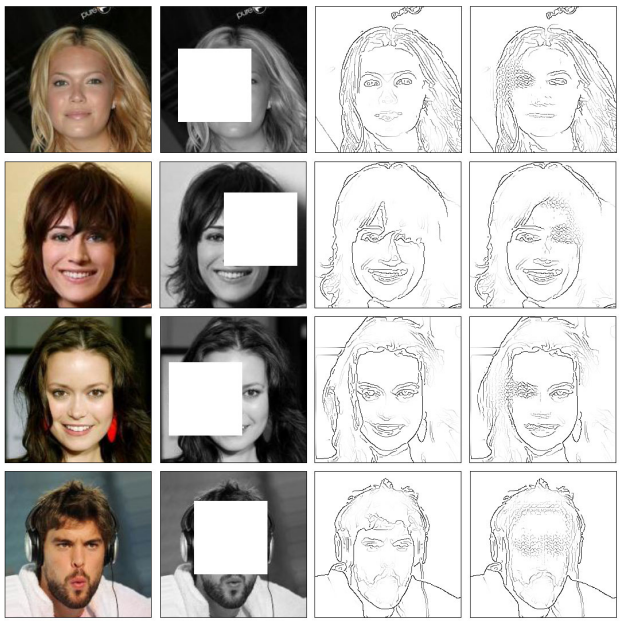
## B.2. Comprehensive Results

Tables 6 and 7 shows the performance of our model compared to existing methods over the datasets CelebA and Paris StreetView respectively. Our method produces noticeably better results. Note that we did not include values for PConv over the CelebA and Paris StreetView datasets as the source code was not available at the time of writing. Figures 13, and 14 display these results graphically. Additional inpainting results of our proposed model are shown in figures 15, 16, and 17.

Our source code, pre-trained models, and more results are available at:
https://github.com/knazeri/edge-connect

## C. Alternative Edge Generating Systems

In drawing, an edge is a boundary that separates two areas. A thick line brings the shape forward thin line indicates a plane receding into the background. In other words edges create a sense of distance and are not just about lines. Here we use HED [48] as an alternative edge detection system. Edges produced with HED, are of varying thickness, and pixels can have intensities ranging between 0 and 1. We noticed that it is possible to create edge maps that look eerily similar to human sketches by performing element-wise multiplication on Canny and HED edge maps. We compare the quantitative results between Canny and a combination of HED and Canny edges (*i.e.* HED⊙Canny). Generated images based on the combined edges gave the best performance. However, our generator $G_1$ is unable to generate these type of edges accurately during training. Table 5 shows $G_1$ trained on HED⊙Canny had the poorest performance out of all methods despite its ground truth counterpart achieving the best performance. These results suggest that better edge detectors result in better inpainting, however, effectively drawing those edges remains an open question in our research. Figure 12 shows the results of $G_1$ trained using hybrid edges.



(a) (b) (c) (d)

Figure 12: Generated edges by $G_1$ trained using hybrid (HED⊙Canny) edges. Images are best viewed in color. (a) Original Image. (b) Image with Masked Region. (c) Ground Truth Edges. (d) Generated Edges.

| | | Hybrid | | Canny | |
|---|---|---|---|---|---|
| | **Mask** | $G_1$ | GT | $G_1$ | GT |
| $\ell_1$ (%)$^\dagger$ | 0-10% | 0.31 | 0.23 | 0.29 | 0.25 |
| | 10-20% | 0.79 | 0.55 | 0.76 | 0.59 |
| | 20-30% | 1.42 | 0.93 | 1.38 | 1.00 |
| | 30-40% | 2.19 | 1.35 | 2.13 | 1.45 |
| | 40-50% | 3.10 | 1.82 | 3.03 | 1.97 |
| | 50-60% | 4.95 | 2.61 | 4.89 | 2.88 |
| SSIM$^\star$ | 0-10% | 0.985 | 0.990 | 0.985 | 0.988 |
| | 10-20% | 0.959 | 0.978 | 0.961 | 0.972 |
| | 20-30% | 0.926 | 0.959 | 0.928 | 0.951 |
| | 30-40% | 0.886 | 0.940 | 0.890 | 0.930 |
| | 40-50% | 0.841 | 0.920 | 0.846 | 0.906 |
| | 50-60% | 0.767 | 0.891 | 0.771 | 0.872 |
| PSNR$^\star$ | 0-10% | 39.24 | 42.43 | 39.60 | 41.77 |
| | 10-20% | 33.26 | 37.48 | 33.51 | 36.81 |
| | 20-30% | 29.80 | 34.65 | 30.02 | 34.00 |
| | 30-40% | 27.21 | 32.59 | 27.39 | 31.92 |
| | 40-50% | 25.12 | 30.87 | 25.28 | 30.21 |
| | 50-60% | 22.03 | 28.49 | 22.11 | 27.68 |
| FID$^\dagger$ | 0-10% | 0.22 | 0.11 | 0.20 | 0.13 |
| | 10-20% | 0.56 | 0.24 | 0.53 | 0.31 |
| | 20-30% | 1.13 | 0.41 | 1.08 | 0.57 |
| | 30-40% | 1.90 | 0.61 | 1.80 | 0.88 |
| | 40-50% | 2.99 | 0.83 | 2.82 | 1.25 |
| | 50-60% | 5.67 | 1.14 | 5.30 | 1.79 |

Table 5: Comparison of quantitative results between Hybrid (HED⊙Canny) and Canny edges over CelebA. Statistics are shown for generated edges ($G_1$) and ground truth edges (GT). $^\dagger$Lower is better. $^\star$Higher is better.

| | Mask | CA | GLCIC | Ours | Canny |
|---|---|---|---|---|---|
| $\ell_1$ (%)$^\dagger$ | 0-10% | 1.33 | 0.91 | **0.29** | 0.25 |
| | 10-20% | 2.48 | 2.53 | **0.76** | 0.59 |
| | 20-30% | 3.98 | 4.67 | **1.38** | 1.00 |
| | 30-40% | 5.64 | 6.95 | **2.13** | 1.45 |
| | 40-50% | 7.35 | 9.18 | **3.03** | 1.97 |
| | 50-60% | 9.21 | 11.21 | **4.89** | 2.88 |
| | Fixed | 2.80 | 3.83 | **2.39** | 1.34 |
| SSIM$^\star$ | 0-10% | 0.947 | 0.947 | **0.985** | 0.988 |
| | 10-20% | 0.888 | 0.865 | **0.961** | 0.972 |
| | 20-30% | 0.819 | 0.773 | **0.928** | 0.951 |
| | 30-40% | 0.750 | 0.689 | **0.890** | 0.930 |
| | 40-50% | 0.678 | 0.609 | **0.846** | 0.906 |
| | 50-60% | 0.614 | 0.560 | **0.771** | 0.872 |
| | Fixed | 0.882 | 0.847 | **0.891** | 0.944 |
| PSNR$^\star$ | 0-10% | 31.16 | 30.24 | **39.60** | 41.77 |
| | 10-20% | 25.32 | 24.09 | **33.51** | 36.81 |
| | 20-30% | 22.09 | 20.71 | **30.02** | 34.00 |
| | 30-40% | 19.94 | 18.50 | **27.39** | 31.92 |
| | 40-50% | 18.41 | 17.09 | **25.28** | 30.21 |
| | 50-60% | 17.18 | 16.24 | **22.11** | 27.68 |
| | Fixed | 25.34 | 22.13 | **25.49** | 31.24 |
| FID$^\dagger$ | 0-10% | 3.24 | 16.84 | **0.20** | 0.13 |
| | 10-20% | 13.12 | 58.74 | **0.53** | 0.31 |
| | 20-30% | 29.47 | 102.97 | **1.08** | 0.57 |
| | 30-40% | 47.55 | 136.47 | **1.80** | 0.88 |
| | 40-50% | 68.40 | 163.95 | **2.82** | 1.25 |
| | 50-60% | 76.70 | 167.07 | **5.30** | 1.79 |
| | Fixed | 1.90 | 25.21 | **1.90** | 0.74 |

Table 6: Comparison of quantitative results over CelebA with CA [53], GLCIC [22], Ours (end-to-end), Ours with Canny edges ($G_2$ only). The best result of each row is boldfaced except for Canny. $^\dagger$Lower is better. $^\star$Higher is better.

| | Mask | CA | GLCIC | Ours | Canny |
|---|---|---|---|---|---|
| $\ell_1$ (%)$^\dagger$ | 0-10% | 0.75 | 0.86 | **0.43** | 0.40 |
| | 10-20% | 2.10 | 2.20 | **1.09** | 0.90 |
| | 20-30% | 3.80 | 3.86 | **1.91** | 1.48 |
| | 30-40% | 5.53 | 5.58 | **2.82** | 2.07 |
| | 40-50% | 7.23 | 7.34 | **3.94** | 2.77 |
| | 50-60% | 9.06 | 9.02 | **5.87** | 3.79 |
| | Fixed | 3.22 | 3.23 | **2.77** | 1.71 |
| SSIM$^\star$ | 0-10% | 0.964 | 0.949 | **0.975** | 0.977 |
| | 10-20% | 0.905 | 0.878 | **0.938** | 0.949 |
| | 20-30% | 0.835 | 0.800 | **0.892** | 0.918 |
| | 30-40% | 0.766 | 0.724 | **0.842** | 0.886 |
| | 40-50% | 0.695 | 0.648 | **0.784** | 0.850 |
| | 50-60% | 0.625 | 0.588 | **0.700** | 0.804 |
| | Fixed | 0.847 | 0.840 | **0.860** | 0.909 |
| PSNR$^\star$ | 0-10% | 32.45 | 30.46 | **36.31** | 37.38 |
| | 10-20% | 26.09 | 25.72 | **31.23** | 33.38 |
| | 20-30% | 22.80 | 22.90 | **28.26** | 31.04 |
| | 30-40% | 20.74 | 21.02 | **26.05** | 29.36 |
| | 40-50% | 19.35 | 19.66 | **24.20** | 27.85 |
| | 50-60% | 18.17 | 18.71 | **21.73** | 25.92 |
| | Fixed | 23.68 | 24.07 | **25.23** | 29.62 |
| FID$^\dagger$ | 0-10% | 2.26 | 6.50 | **0.44** | 0.31 |
| | 10-20% | 9.10 | 18.77 | **1.20** | 0.68 |
| | 20-30% | 20.62 | 35.66 | **2.49** | 1.24 |
| | 30-40% | 34.31 | 53.53 | **4.35** | 1.89 |
| | 40-50% | 49.80 | 70.36 | **7.20** | 2.78 |
| | 50-60% | 55.78 | 69.95 | **13.98** | 4.12 |
| | Fixed | 7.26 | 7.18 | **4.57** | 3.24 |

Table 7: Comparison of quantitative results over Paris StreetView with CA [53], GLCIC [22], Ours (end-to-end), Ours with Canny edges ($G_2$ only). The best result of each row is boldfaced except for Canny. $^\dagger$Lower is better. $^\star$Higher is better.

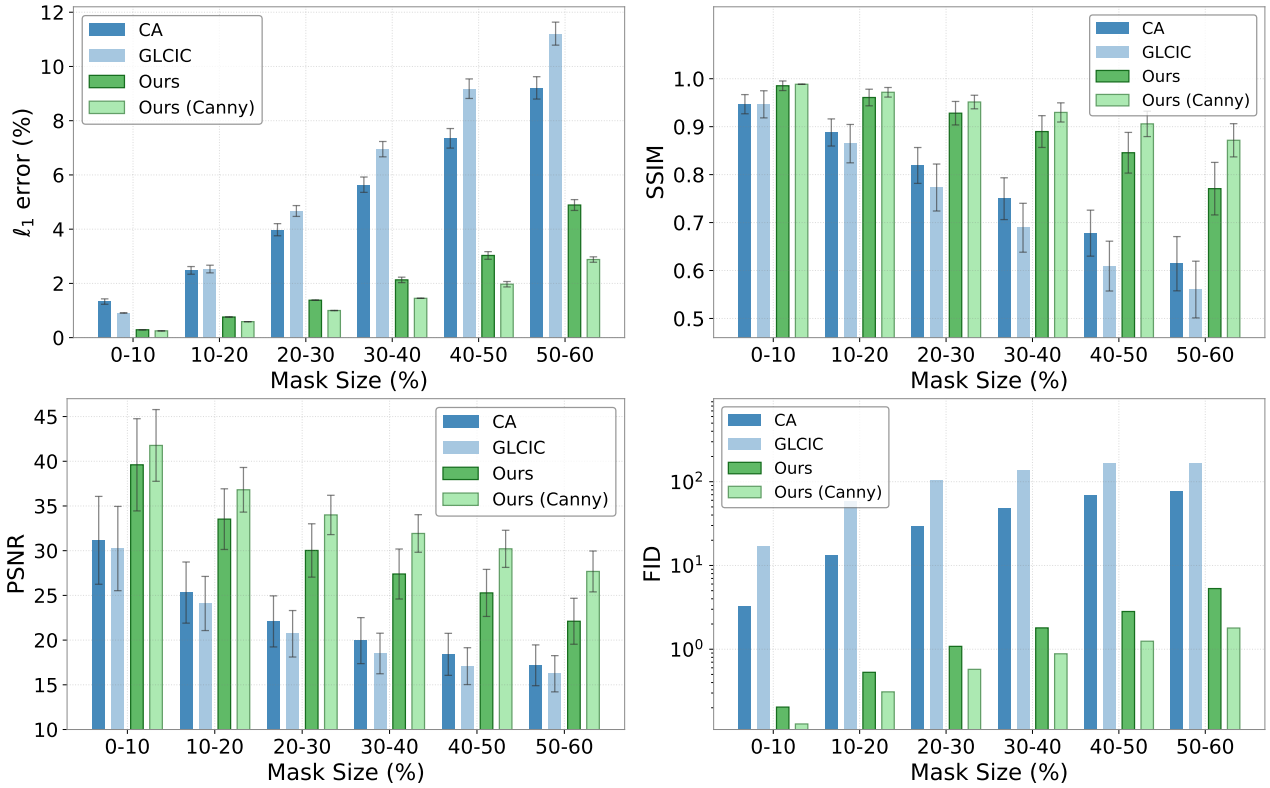



Figure 13: Effect of relative mask sizes on $\ell_1$, SSIM, PSNR, and FID on the CelebA dataset.

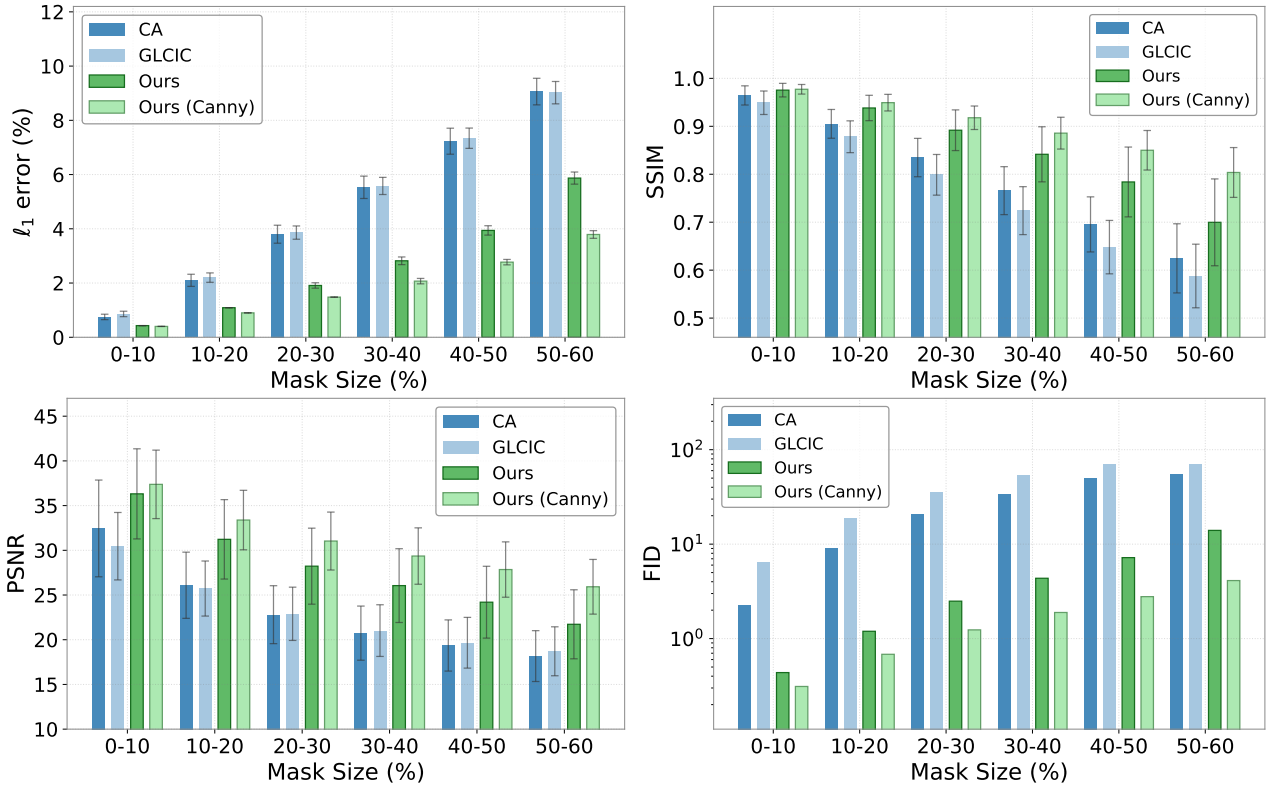



Figure 14: Effect of relative mask sizes on $\ell_1$, SSIM, PSNR, and FID on the Paris StreetView dataset.

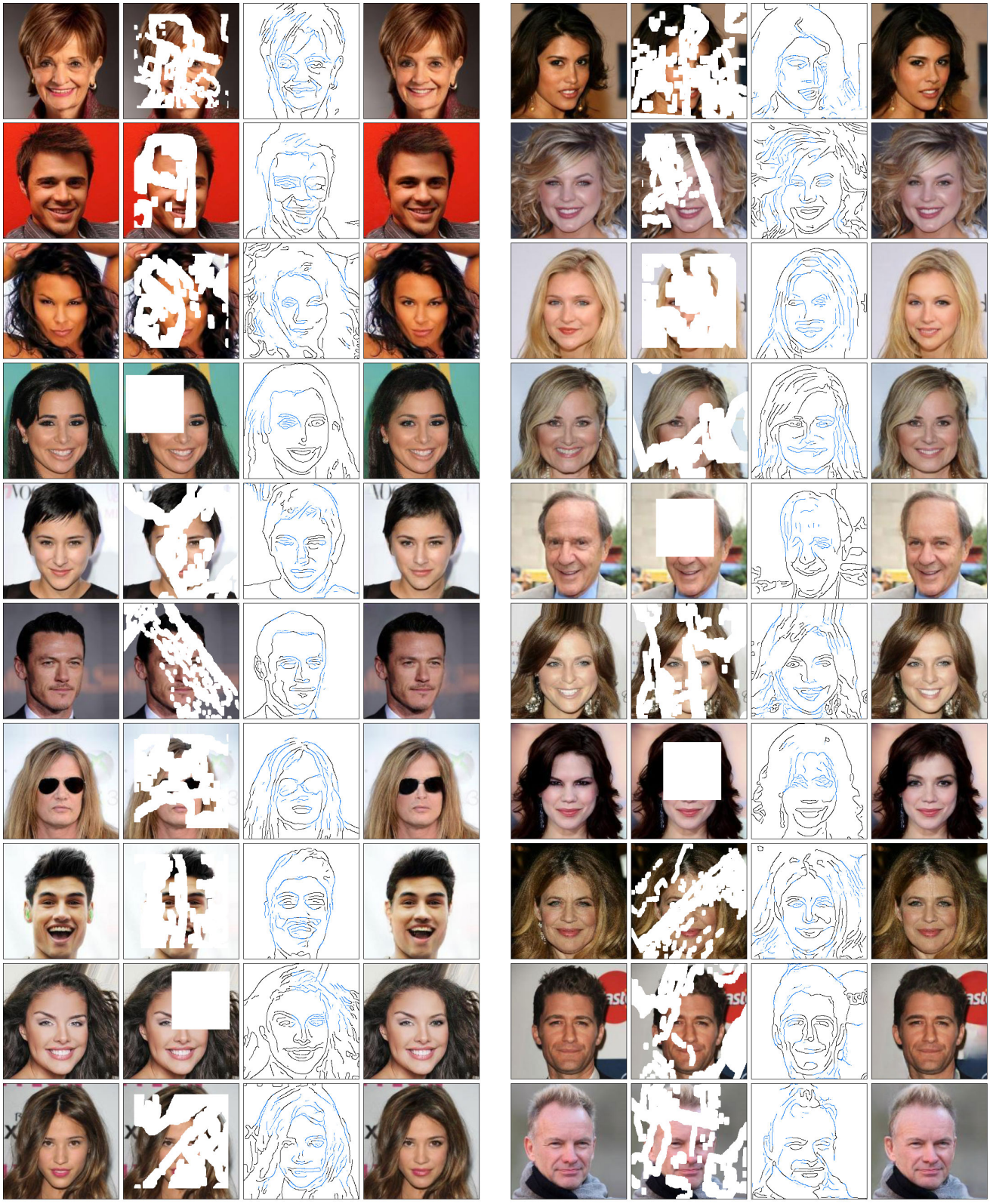



Figure 15: Sample of results with CelebA dataset. Images are best viewed in color. (a) Original Image. (b) Input Image. (c) Generated Edges (Blue) and Ground Truth Edges (Black). (d) Generated Result.

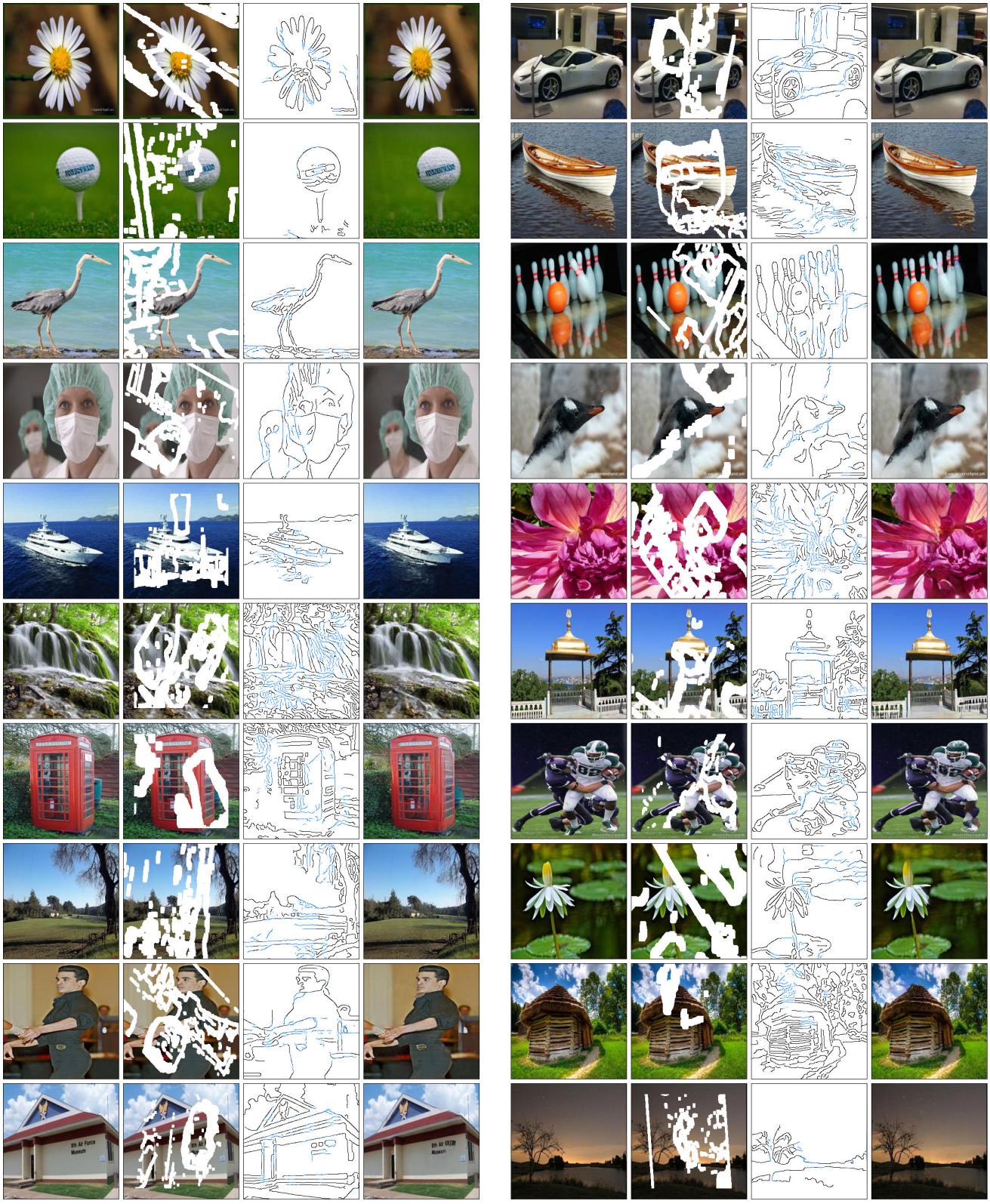



Figure 16: Sample of results with Places2 dataset. Images are best viewed in color. (a) Original Image. (b) Input Image. (c) Generated Edges (Blue) and Ground Truth Edges (Black). (d) Generated Result.

(a) (b) (c) (d) (a) (b) (c) (d)

Figure 17: Sample of results with Paris StreetView dataset. Images are best viewed in color. (a) Original Image. (b) Input Image. (c) Generated Edges (Blue) and Ground Truth Edges (Black). (d) Generated Result.