Contextual Residual Aggregation for Ultra High-Resolution Image Inpainting

Zili Yi Qiang Tang Shekoofeh Azizi Daesik Jang Zhan Xu

Huawei Technologies Canada Co. Ltd.

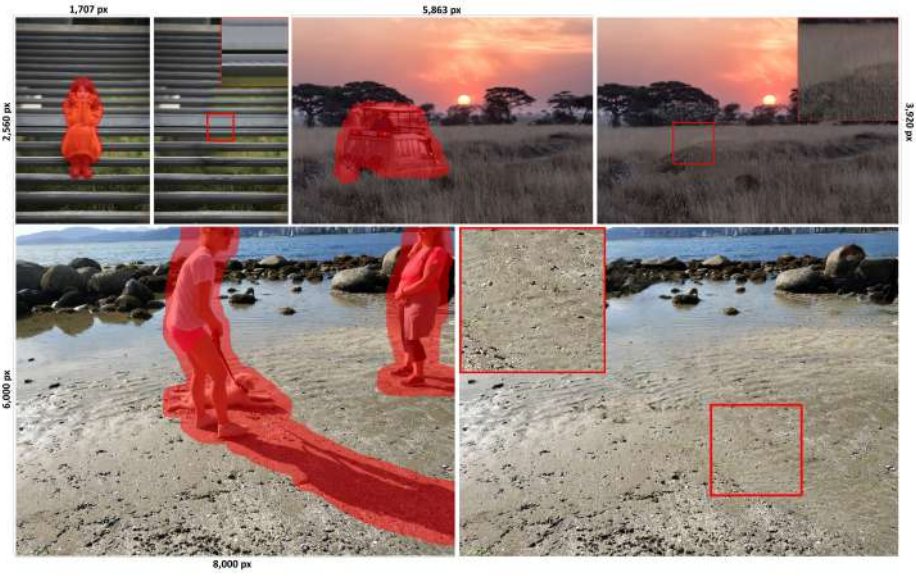


Fig. 1: Inpainting results on ultra high-resolution images.

Abstract. Recently data-driven image inpainting methods have made inspiring progress, impacting fundamental image editing tasks such as object removal and damaged image repairing. These methods are more effective than classic approaches, however, due to memory limitations they can only handle low-resolution inputs, typically smaller than 1K. Meanwhile, the resolution of photos captured with mobile devices increases up to 8K. Naive up-sampling of the low-resolution inpainted result can merely yield a large yet blurry result. Whereas, adding a high-frequency residual image onto the large blurry image can generate a sharp result, rich in details and textures. Motivated by this, we propose a Contextual Residual Aggregation (CRA) mechanism that can produce high-frequency residuals for missing contents by weighted aggregating residuals from contextual patches, thus only requiring a low-resolution prediction from the network. Since convolutional layers of the neural network only need to operate on low-resolution inputs and

outputs, the cost of memory and computing power is thus well suppressed. Moreover, the need for high-resolution training datasets is alleviated. In our experiments, we train the proposed model on small images with resolutions 512×512 and perform inference on high-resolution images, achieving compelling inpainting quality. Our model can inpaint images as large as 8K with considerable hole sizes, which is intractable with previous learning-based approaches. We further elaborate on the light-weight design of the network architecture, achieving real-time performance on 2K images on a GTX 1080 Ti GPU. Codes are available at: Atlas200dk/sample-imageinpainting-HiFill.

Keywords: Image Inpainting; Image Completion; Ultra high-resolution; Generative Adversarial Network; Contextual Residual Aggregation; Contextual Attention; Light-Weight Gated Convolution

1 Introduction

Smartphone users are interested to manipulate their photographs in any form of altering object positions, removing unwanted visual elements, or repairing damaged images. These tasks require automated image inpainting, which aims at restoring lost or deteriorated parts of an image given a corresponding mask. Inpainting has been an active research area for the past few decades, however, due to its inherent ambiguity and the complexity of natural images, general image inpainting remains challenging. High-quality inpainting usually requires generating visually realistic and semantically coherent content to fill the hole regions. Existing methods for image hole filling can be categorized into three groups. The first category which we call *"fill through copying"* attempts to explicitly borrow contents or textures from surroundings to fill the missing regions. An example is diffusion-based [1, 2] methods which propagate local image appearance surrounding the target holes based on the isophote direction field. Another stream is relying on texture synthesis techniques, which fills the hole by both extending and borrowing textures from surrounding regions [3, 4, 5, 6, 7]. Patch-based algorithms like [4, 8, 9, 6] progressively fill pixels in the hole by searching the image patches from background regions that are the most similar to the pixels along the hole boundaries.

The second group attempts to *"fill through modeling"* and hallucinates missing pixels in a data-driven manner with the use of large external databases. These approaches learn to model the distribution of the training images and assume that regions surrounded by similar contexts likely to possess similar contents [10, 11, 12, 13, 14, 15]. For instance, PixelRNN [12] uses a two-dimensional Recurrent Neural Network (RNN) to model the pixel-level dependencies along two spatial dimensions. More general idea [10, 15] is to train an encoder-decoder convolutional network to model the 2-dimensional spatial contents. Rather than modeling the raw pixels, [11, 14] train a convolutional network to model image-wide edge structure or foreground object contours, thus enabling auto-completion of the edge or contours. These techniques are effective when they find an example image with sufficient visual similarity to the query, but will easily fail if the database does not have similar examples. To overcome the limitation of copying-based or modeling-based methods, the third group of approaches attempts to combine the two [16, 17, 18, 19, 20, 21]. These methods learn to model the image

distribution in a data-driven manner, and in the meantime, they develop mechanisms to explicitly borrow patches/features from background regions. [20] introduces a novel contextual attention layer that enables borrowing features from distant spatial locations. [21] further extends the contextual attention mechanism to multiple scales and all the way from feature-level to image-level. [17] employs the patch-swap layer that propagates the high-frequency texture details from the boundaries to hole regions.

Most learning-based approaches belong to the second or third group. Compared to traditional methods, these techniques have strong ability to learn adaptive and high-level features of disparate semantics and thus are more adept in hallucinating visually plausible contents especially when inpainting structured images like faces [10, 12, 17, 19, 20, 21], objects [11, 13, 14, 15], and natural scenes [10, 17, 19, 20]. Since existing methods employ convolutional layers directly on the original input, the memory usage could become extremely high and intractable when the input size is up to 8K. Another issue is that the quality deteriorates rapidly when hole size increases with image size. Even if the training is feasible, access to large amounts of high-resolution training data would be tedious and expensive.

To resolve these issues, we propose a novel Contextual Residual Aggregation (CRA) mechanism to enable the completion of ultra high-resolution images with limited resources. In specific, we use a neural network to predict a low-resolution inpainted result and up-sample it to yield a large blurry image. Then we produce the high-frequency residuals for in-hole patches by aggregating weighted high-frequency residuals from contextual patches. Finally, we add the aggregated residuals to the large blurry image to obtain a sharp result. Since the network only operates on low-resolution images, the cost of memory and computing time is significantly reduced. Moreover, as the model can be trained with low-resolution images, the need for high-resolution training datasets is alleviated. Furthermore, we introduce other techniques including slim and deep layer configuration, attention score sharing, multi-scale attention transfer, and Light-Weight Gated Convolutions (LWGC) to improve the inpainting quality, computation, and speed. Our method can inpaint images as large as 8K with satisfying quality, which cannot be handled by prior learning-based approaches. Exemplar results are shown in Figure 1. The contributions of the paper are summarized as follows:

- We design a novel and efficient Contextual Residual Aggregation (CRA) mechanism that enables ultra high-resolution inpainting with satisfying quality. The mechanism enables large images (up to 8K) with considerable hole sizes (up to 25%) to be inpainted with limited memory and computing resources, which is intractable for prior methods. Also, the model can be trained on small images and applied on large images, which significantly alleviates the requirements for high-resolution training datasets.
- We develop a light-weight model for irregular hole-filling that can perform real-time inference on images of 2K resolutions on a NVIDIA GTX 1080 Ti GPU, using techniques including slim and deep layer configuration, attention score sharing, and Light Weight Gated Convolution (LWGC).
- We use attention transfer at multiple abstraction levels which enables filling holes by weighted copying features from contexts at multiple scales, improving the in-

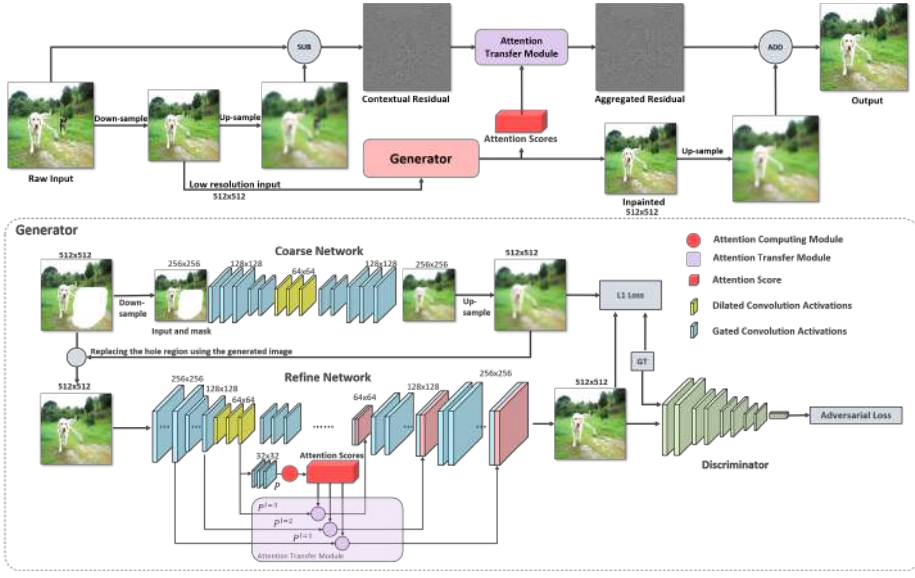


Fig. 2: The overall pipeline of the method: (top) CRA mechanism, (bottom) the architecture of the generator.

painting quality over existing methods by a certain margin even when tested on low-resolution images.

2 Related Works

2.1 Irregular Hole-filling & Modified Convolutions

Vanilla convolutions are intrinsically troublesome for irregular hole-filling because convolutional filters treat all pixels the same as valid ones, causing visual artifacts such as color inconsistency, blurriness, and boundary artifacts. Partial convolution [16] is proposed to handle irregular holes, where the convolution is masked and re-normalized to be conditioned on valid pixels. Gated convolution [19] generalizes the partial convolution idea by providing a learnable dynamic feature selection mechanism for each channel and at each spatial location, achieving better visual performance. Here, we further improve the gated convolution through a lightweight design to improve efficiency.

2.2 Contextual Attention

Contextual attention [20] is proposed to allow long-range spatial dependencies during inpainting, which enables borrowing pixels from distant locations to fill missing regions. The contextual attention layer has two phases: “match” and “attend”. In the

"match" phase, the attention scores are computed by obtaining region affinity between patches inside and those outside the holes. In the "attend" phase, holes are filled by copying and aggregating patches from weighted contexts by the attention scores. [21] extends this idea by using a pyramid of contextual attention at multiple layers. In contrast to [21], we only compute the attention scores once and reuse them at multiple abstraction levels, which leads to fewer parameters and less computation.

2.3 Image Residuals

The difference between an image and the blurred version of itself represents the high-frequency image [22, 23]. Early works use the difference obtained by Gaussian blurring for low-level image processing tasks like edge detection, image quality assessment, and feature extraction [24, 25, 26]. We employ this concept to decompose the input image into low-frequency and high-frequency components. The low-frequency component is obtained through averaging neighboring pixels, whereas the high-frequency component (i.e. image residuals) is obtained by subtracting the original image with its low-frequency component.

3 Method

3.1 The Overall Pipeline

Figure 2 illustrates the overall pipeline of the proposed CRA mechanism where the generator is the only trainable component in the framework. Given a high-resolution input image, we first down-sample the image to 512×512 and then up-sample it to obtain a blurry large image of the same size as the raw input (Section 4.1). The height and width of the image are not necessary to be equal but must be multiples of 512. The generator takes the low-resolution image and fills the holes. Meanwhile, the attention scores are calculated by the Attention Computing Module (ACM) of the generator (Section 3.2). Also, the contextual residuals are computed by subtracting the large blurry image from the raw input, and the aggregated residuals in the mask region are then calculated from the contextual residuals and attention scores through an Attention Transfer Module (ATM) (Section 3.2). Finally, adding the aggregated residuals to the up-sampled inpainted result generates a large sharp output in the mask region while the area outside mask is simply a copy of the original raw input.

3.2 Contextual Residual Aggregation (CRA)

Filling the missing region by using contextual information [17, 18, 27], and contextual attention mechanism [20] has been proposed previously. Similarly, we deploy the CRA mechanism to borrow information from contextual regions. However, the CRA mechanism borrows from contexts not only features but also residuals. In particular, we adopt the idea of contextual attention [20] in calculating attention scores by obtaining region affinity between patches inside/outside missing regions. Thus contextually relevant features and residuals outside can be transferred into the hole. Our mechanism involves two key modules: Attention Computing Module and Attention Transfer Module.

Attention Computing Module (ACM) The attention scores are calculated based on region affinity from a high-level feature map (denoted as P in Figure 2). P is divided into patches and ACM calculates the cosine similarity between patches inside and outside missing regions:

$$c_{i,j} = \langle \frac{p_i}{\|p_i\|}, \frac{p_j}{\|p_j\|} \rangle \tag{1}$$

where p_i is the i^{th} patch extracted from P outside mask, p_j is the j^{th} patch extracted from P inside the mask. Then softmax is applied on the similarity scores to obtain the attention scores for each patch:

$$s_{i,j} = \frac{e^{c_{i,j}}}{\Sigma_{i=1}^{N} e^{c_{i,j}}} \tag{2}$$

where N is the number of patches outside the missing hole. In our framework, each patch size is 3×3 and P is 32×32, thus a total number of 1024 patches can be extracted. In practice, the number of in-hole patches could vary for different hole sizes. We uniformly use a matrix of 1024×1024 to save affinity scores between any possible pair of patches, though only a fraction of them are useful.

Attention Transfer Module (ATM) After obtaining the attention scores from P, the corresponding holes in the lower-level feature maps (P^l) can be filled with contextual patches weighted by the attention scores:

$$p_j^l = \Sigma_{i=1}^{N} s_{i,j} p_i^l \tag{3}$$

where $l \in 1, 2, 3$ is the layer number and p_i^l is the i^{th} patch extracted from P^l outside masked regions, and p_j^l is the j^{th} patch to be filled inside masked regions. N indicates the number of contextual patches (background). After calculating all in-hole patches, we can finally obtain a filled feature P^l. As the size of feature maps varies by layer, the size of patches should vary accordingly. Assuming the size of the feature map is 128^2 and the attention scores are computed from 32^2 patches, then the patch sizes should be greater or equal to $(128/32)^2 = 4^2$ so that all pixels can be covered. If the patch sizes are greater than 4×4, then certain pixels are overlapped, which is fine as the following layers of the network can learn to adapt.

Multi-scale attention transfer and score sharing. In our framework, we apply attention transfer multiple times with the same set of attention scores (Figure 2). The sharing of attention scores leads to fewer parameters and better efficiency in terms of memory and speed.

Residual Aggregation The target of Residual Aggregation is to calculate residuals for the hole region so that sharp details of the missing contents could be recovered. The residuals for the missing contents can be calculated by aggregating the weighted contextual residuals obtained from previous steps:

$$R_j = \Sigma_{i=1}^{N} s_{i,j} R_i \tag{4}$$

where R is the residual image and R_i is the i^{th} patch extracted from contextual residual image outside the mask, and R_j is j^{th} patch to be filled inside the mask. The patch sizes are properly chosen to exactly cover all pixels without overlapping, to ensure the filled residuals being consistent with surrounding regions. Once the aggregated residual image is obtained, we add it to the up-sampled blurry image of the generator, and obtain a sharp result (Figure 2).

3.3 Architecture of Generator

The network architecture of the generator is shown in Figure 2. We use a two-stage coarse-to-fine network architecture where the coarse network hallucinates rough missing contents, and the refine network predicts finer results. The generator takes an image and a binary mask indicating the hole regions as input and predicts a completed image. The input and output sizes are expected to be 512×512. In order to enlarge the perceptive fields and reduce computation, inputs are down-sampled to 256×256 before convolution in the coarse network, different from the refine network who operates on 512×512. The prediction of the coarse network is naively blended with the input image by replacing the hole region of the latter with that of the former as the input to the refine network. Refine network computes contextual attention scores with a high-level feature map and performs attention transfer on multiple lower-level feature maps, thus distant contextual information can be borrowed at multiple abstraction levels. We also adopt dilated convolutions [10] in both coarse and refine networks to further expand the size of the receptive fields. To improve the computational efficiency, our inpainting network is designed in a slim and deep fashion, and the LWGC is applied for all layers of the generator. Other implementation considerations include: (1) using 'same' padding and ELUs [28] as activation for all convolution layers, (2) removing batch normalization layer as they deteriorate color coherency [10].

3.4 Light Weight Gated Convolution

Gated Convolutions (GC) [19] leverages the art of irregular hole inpainting. However, GC almost doubles the number of parameters and processing time in comparison to vanilla convolution. In our network, we proposed three modified versions of GC called Light Weight Gated Convolutions (LWGC), which reduces the number of parameters and processing time while maintaining the effectiveness. The output of the original GC can be expressed as:

$$\begin{aligned} G &= \text{conv}(W_g, I) \\ F &= \text{conv}(W_f, I) \\ O &= \sigma(G) \odot \psi(F) \end{aligned} \tag{5}$$

where σ is Sigmoid function thus the output values are within $[0, 1]$. ψ is an activation function which are set to ELU in our experiments. w_g and w_f are two different set of convolutional filters, which are used to compute the gates and features respectively. GC enables the network to learn a dynamic feature selection mechanism. The three

Table 1: Number of parameters needed to compute gates

Method	Parameters Calculation	$H_k, W_k = 3$ $C_{in}, C_{out} = 32$
GC [19]	$H_k \times W_k \times C_{in} \times C_{out}$	9216
LWGC^{ds}	$H_k \times W_k \times C_{in} + C_{in} \times C_{out}$	1312
LWGC^{pw}	$C_{in} \times C_{out}$	1024
LWGC^{sc}	$H_k \times W_k \times C_{in} \times 1$	288

variations of LWGC that we propose are named as: depth-separable LWGC (LWGC^{ds}), pixelwise LWGC (LWGC^{pw}), and single-channel LWGC (LWGC^{sc}). They differ by the computation of the gate branch, G:

$$G = \text{conv}^{depth-separable}(W_g, I) \tag{6}$$

$$G = \text{conv}^{pixelwise}(W_g, I) \tag{7}$$

$$G = \text{conv}(W_g, I), \text{ G is single-channel} \tag{8}$$

The depth-separable LWGC employs a depth-wise convolution followed by a 1×1 convolution to compute gates. The pixelwise LWGC uses a pixelwise or 1×1 convolution to compute the gates. The single-channel LWGC outputs a single-channel mask that is broadcast to all feature channels during multiplication. The single-channel mask is similar to partial convolution, whereas the mask of partial convolutions is hard-wired and untrainable, and generates a binary mask instead of a soft mask. Given that the height (H_k) and width (W_k) of kernels, and numbers of input channels (C_{in}) and output channels (C_{out}), we compare the number of parameters needed to calculate gates in Table 1. We used the single-channel LWGC for all layers of the coarse network and depth-separable or pixelwise LWGC for all layers of the refine network, which has been proved to be equally effective as regular GC but more efficient (Section 4.2).

3.5 Training of the network

Training Losses Without the degradation of performance, we also significantly simplify the training objectives as two terms: the adversarial loss and the reconstruction loss. We use the WGAN-GP loss as our adversarial loss [29], which enforces global consistency in the second-stage refinement network. The discriminator and generator are alternatively trained with the losses defined in Equation 9 and Equation 10:

$$\begin{aligned} L_d = \mathbb{E}_{\tilde{\mathbf{x}} \in \mathbb{P}_g}[D(\tilde{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \in \mathbb{P}_r}[D(\mathbf{x})] + \\ \sigma \mathbb{E}_{\hat{\mathbf{x}} \in \mathbb{P}_{\hat{\mathbf{x}}}}[\| \bigtriangledown_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1]^2 \end{aligned} \tag{9}$$

where $D(.)$ is the discriminator output and $G(.)$ is the generator output. $\mathbf{x}$, $\tilde{\mathbf{x}}$, $\hat{\mathbf{x}}$, are real images, generated images, and interpolations between them, respectively. $\mathbb{P}_g$, $\mathbb{P}_r$, $\mathbb{P}_{\hat{\mathbf{x}}}$ are the corresponding distributions of them respectively.

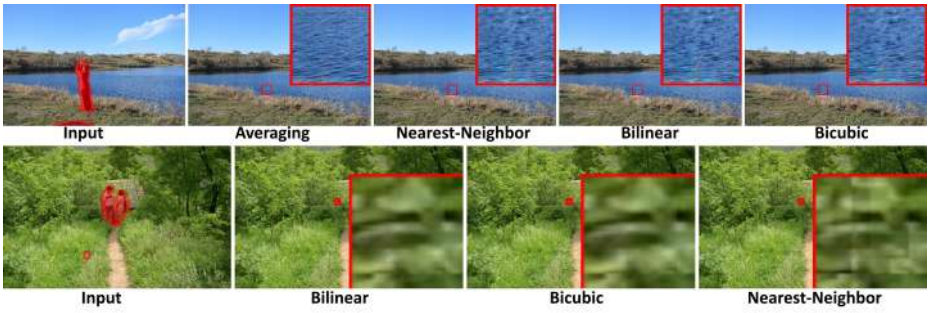


Fig. 3: Comparing down-sampling and up-sampling operators: (top) using Bilinear up-sampling along with Averaging down-sampling generates more coherent textures with the surroundings. (bottom) using the Averaging down-sampling along with Nearest Neighbor produces tiling artifacts while Bilinear and Bicubic up-sampling perform equally well.

$$L_{adv} = -\mathbb{E}_{\tilde{\mathbf{x}} \in \mathbb{P}_g}[D(\tilde{\mathbf{x}})] \tag{10}$$

We also add the $L1$ loss to force the consistency of the prediction with the original image. In contrast to [20], we avoid the computationally expensive spatially-discounted reconstruction loss. For simplicity, we just assign a smaller constant weight for the reconstruction loss of all in-hole pixels. The reconstruction loss is thus written as:

$$L_{in-hole} = |G(\mathbf{x}, \mathbf{m}) - \mathbf{x}| \odot \mathbf{m} \tag{11}$$

$$L_{context} = |G(\mathbf{x}, \mathbf{m}) - \mathbf{x}| \odot (1 - \mathbf{m}) \tag{12}$$

$$L_{rec} = \alpha_1 L_{in-hole} + \alpha_2 L_{context} \tag{13}$$

where α_1 and α_2 are coefficients for the in-hole term and contextual term ($\alpha_1 = 1$, and $\alpha_2 = 1.2$). The coarse network is trained with the reconstruction loss explicitly, while the refinement network is trained with a weighted sum of the reconstruction and GAN losses. The coarse network and refine network are trained simultaneously with merged losses as shown in Equation 14.

$$L_g = L_{rec} + \beta L_{adv} \tag{14}$$

where β is the coefficient for adversarial loss ($\beta = 10^{-4}$).

Random Mask Generation To diversify the inpainting masks, we use two methods to generate irregular masks on-the-fly during training. The first one is [16], which simulates tears, scratches, spots or manual erasing with brushes. The second approach generates masks by randomly manipulating the real object shape templates, accounting for

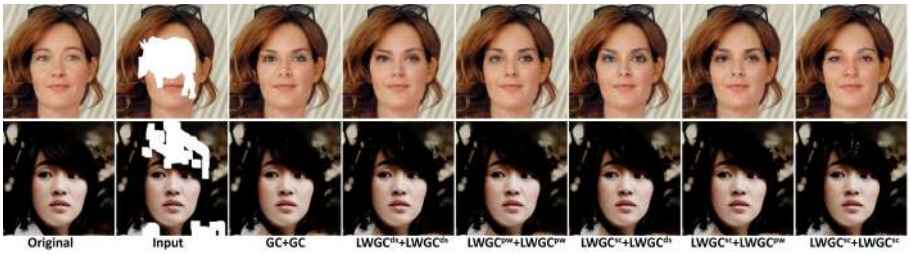


Fig. 4: Comparisons of different Gated Convolutions configurations. For example, the notation of LWGC^{sc}+LWGC^{ds} means: the coarse network uses single-channel LWGC and the refine network uses depth-separable LWGC.

the object removal scenario. These shape templates are obtained from object segmentation masks and including a wide range of categories such as single, multiple or crowded objects. We also randomly rotate, flip and scale the templates with a random scale ratio. In practice, the aforementioned two methods can be combined or separated, depending on specific needs.

Training Procedure During training, color values of all images are linearly scaled to $[-1, 1]$ in all experiments, and the mask uses 1 to indicate the hole region and 0 to indicate background. The masked image is constructed as $\mathbf{x} \odot (1 - \mathbf{m})$, where $\mathbf{x}$ is the input image and $\mathbf{m}$ is the binary mask, and $\odot$ represents dot product. Inpainting generator G takes concatenation of the masked image and mask as input, and predicts $\mathbf{y} = G(\mathbf{x}, \mathbf{m})$ of the same size as the input image. The entire training procedure is illustrated in Algorithm 1.

4 Experimental Results

We evaluate the proposed method on three datasets including Places2 [30], CelebA-HQ [31], and DIV2K [32]. Our model is trained on two NVIDIA 1080 Ti GPUs with images of resolution 512×512 with batch size of 8. For DIV2K and CelebA-HQ, images are down-sampled to 512×512. For Places2, images are randomly cropped to 512×512. After training, we test the models on images of various resolutions of 512 to 8K on a GPU. The final model has a total of 2.7M parameters and implemented on TensorFlow v1.13 with CUDNN v7.6 and CUDA v10.0.

4.1 Analysis of CRA Design

As shown in Figure 2, the CRA mechanism involves one down-sampling and two up-sampling operations outside of the generator. Choosing different methods for down-sampling and up-sampling may affect the final results. To explore this, we experimented

Algorithm 1: Training of our proposed network

initialization;
while *G has not converged* **do**
  for *i = 1,...,5* **do**
    Sampling batch images $\mathbf{x}$ from training data;
    Generating random masks $\mathbf{m}$ for $\mathbf{x}$;
    Getting inpainted $\mathbf{y} \leftarrow G(\mathbf{x}, \mathbf{m})$;
    Pasting back $\tilde{\mathbf{x}} \leftarrow \mathbf{y} \odot \mathbf{m} + \mathbf{x} \odot (1 - \mathbf{m})$;
    Sampling a random number $\alpha \in U[0, 1]$;
    Getting interpolation $\hat{\mathbf{x}} \leftarrow (1 - \alpha)\mathbf{x} + \alpha\tilde{\mathbf{x}}$;
    Updating the discriminator D with loss L_d;
  end
  Sampling batch images $\mathbf{x}$ from training data;
  Generating random masks $\mathbf{m}$ for $\mathbf{x}$;
  Getting Inpainted $\mathbf{y} \leftarrow G(\mathbf{x}, \mathbf{m})$;
  Pasting back $\tilde{\mathbf{x}} \leftarrow \mathbf{y} \odot \mathbf{m} + \mathbf{x} \odot (1 - \mathbf{m})$;
  Updating generator G with loss L_g;
end

Table 2: Quantitative evaluation results on Places2 validation set. Note that certain models cause Out-Of-Memory (OOM) error when tested on 2K or 4K images, thus the corresponding cells are left empty.

Image Size	512×512					1024×1024					2048×2048					4096×4096				
Metrics	L1	MS-SSIM	FID	IS	Time	L1	MS-SSIM	FID	IS	Time	L1	MS-SSIM	FID	IS	Time	L1	MS-SSIM	FID	IS	Time
DeepFillV1[20]	6.733	0.8442	7.541	17.56	62ms	7.270	0.8424	10.21	17.69	663ms	–	–	–	–	–	–	–	–	–	–
DeepFillV2[19]	6.050	**0.8848**	4.939	**18.20**	78ms	6.942	0.8784	8.347	17.04	696ms	–	–	–	–	–	–	–	–	–	–
PEN-Net[21]	9.732	0.8280	14.13	14.51	35ms	10.42	0.8128	19.36	12.51	289ms	–	–	–	–	–	–	–	–	–	–
PartialConv[16]	8.197	0.8399	29.32	13.13	35ms	11.19	0.8381	32.20	13.53	110ms	16.19	0.8373	41.23	11.17	920ms	–	–	–	–	–
Global-local[10]	8.617	0.8469	21.27	13.48	53ms	9.232	0.8392	26.23	13.05	219ms	9.308	0.8347	27.09	12.61	219ms	–	–	–	–	–
Ours	**5.439**	0.8840	**4.898**	17.72	**25ms**	**5.439**	**0.8840**	**4.899**	**17.72**	**31ms**	**5.492**	**0.8840**	**4.893**	**17.85**	**37ms**	**5.503**	**0.8840**	**4.895**	**17.81**	**87.3ms**

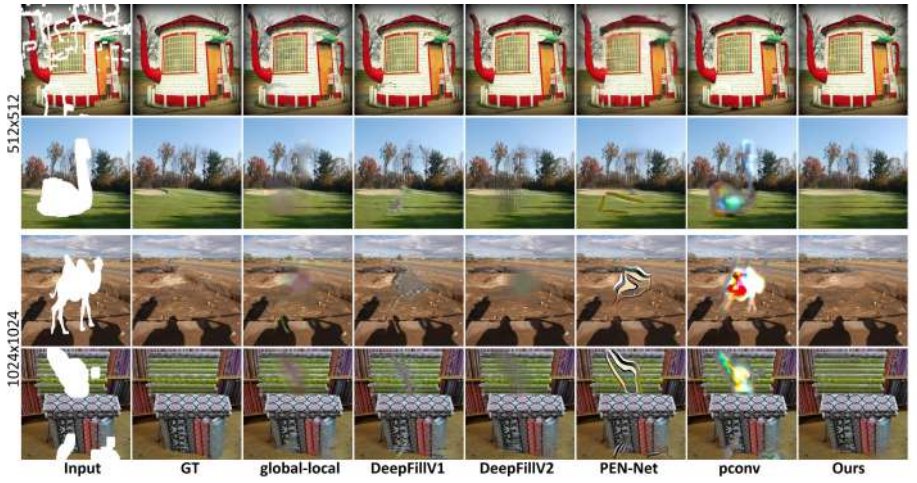


Fig. 5: Qualitative comparisons using 512×512 (top) and 1024×1024 (bottom) images from Places2 validation dataset.

with four down-sampling methods: Nearest-Neighbor, Bilinear, Bicubic, and Averaging. Averaging evenly splits the input into target patches and average all pixels in each patch to obtain a 512×512 image. We also explored three up-sampling methods including Nearest-Neighbor, Bilinear or Bicubic. Note that the two up-sampling operations must be consistent, so we do not consider inconsistent combinations. Experimental results on an HD dataset indicate that Averaging performs the best for down-sampling and Bilinear or Bicubic performs equally well for up-sampling (Figure 3). For simplicity, we use Averaging down-sampling and Bilinear up-sampling.

4.2 Analysis of Light Weight Gated Convolutions

We propose three types of LWGC, which are faster than the original GC. We experimented how they affect inpainting quality and efficiency on the CelebA-HQ dataset to explore the influence of LWGC on the results, by replacing the original GCs with LWGCs for the coarse/refine networks. As shown in Figure 4, the LWGC^{sc}+LWGC^{sc} configuration brings visible artifacts while the other five configurations perform equally well in terms of quality. Considering LWGC^{sc}+LWGC^{pw} requires fewer parameters than the other four, we adopt the LWGC^{sc}+LWGC^{pw} configuration in the generator.

4.3 Comparisons With Learning-based Methods

We compared our method with other state-of-the-art learning-based inpainting methods including Global-local GAN [10], DeepFillV1 [20], DeepFillV2 [19], PEN-Net [21] and Partial Convolution [16]. To make fair comparisons, we attempted to use the same

settings for all experiments, though not fully guaranteed. The official pre-trained DeepFillV1 [20] model was trained for 100M iterations with the batch size of 16 and the global-local GAN [10] was trained for 300K iterations with the batch size of 24. Both of them were trained on 256×256 images with rectangular holes of maximum size 128×128. All the other models were trained for 300K iterations with the batch size of 8 on 512×512 images with irregular holes up to 25% area of the whole image. The original DeepFillV2 [19] model attached a sketch channel to the input to facilitate image manipulation, we simply removed the sketch channel and re-trained the model. For all these methods, no specific post-processing steps are performed other than pasting the filled contents back to the original image.

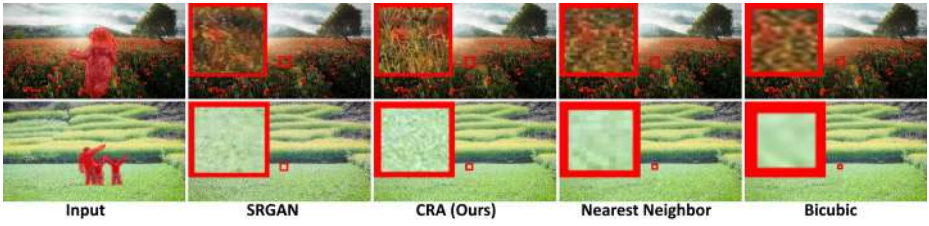


Fig. 6: Comparisons of different super-resolution methods: the red squares area are zoomed-in for more details.

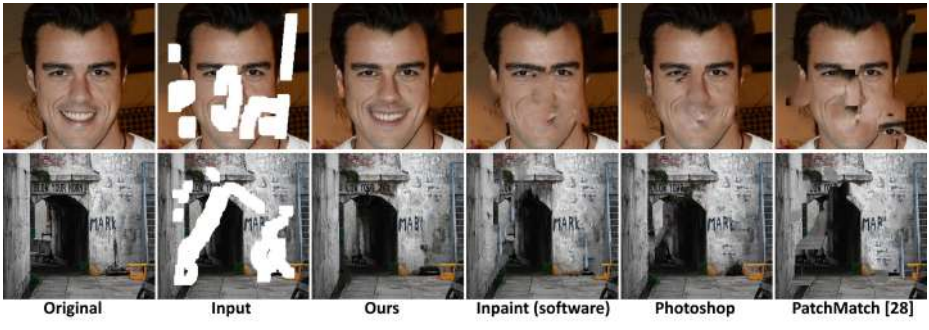


Fig. 7: Comparisons of our method with Inpaint (software), Photoshop content-aware fill and an open-source PatchMatch implementation [33]. The masks for Photoshop and Inpaint are manually drawn, thus not guaranteed to be the same.

Qualitative comparisons Figure 5 shows our model performs equally good or slightly better than previous methods on 512×512 images. Partial convolution [16] and global-local GAN [10] performs well when the mask is small and narrow but exert severe

artifacts when the hole size becomes bigger. Global-local GAN [10] is problematic in maintaining the color consistency of filled contents with surroundings. DeepFillV1 [20] generates plausible results, but occasionally the artifacts inside the hole region are visible, implying its vulnerability to irregular masks. DeepFillV2 [19] generates incoherent textures when the hole size goes up. Nevertheless, when tested on larger images with bigger hole sizes, our model performs consistently good while the inpainting results of other methods deteriorate dramatically (e.g. severe hole-shaped artifacts in Figure 5).

Quantitative comparisons Table 2 reports our quantitative evaluation results in terms of mean $L1$ error, MS-SSIM [34], Inception Score (IS) [35], and Frechet Inception Distance (FID) [36]. It also shows the average inference time per image on a NVIDIA GTX 1080 Ti GPU. These metrics are calculated over all 36,500 images of the Places2 validation set. Each image is assigned a randomly generated irregular mask. To examine the performance on various image sizes, we linearly scale images and masks to various dimensions. Table 2 shows that our proposed model achieves the lowest $L1$ loss and FID on 512×512 images. When the input images are greater than or equal to 1024×1024, our proposed model achieves the best results on all metrics. In addition, the proposed approach significantly outperforms other learning-based methods in terms of speed. In specific, it is 28.6% faster for 512×512 images, 3.5 times faster for 1024×1024 images, and 5.9 times faster for 2048×2048 images than the second-fastest method. Furthermore, the proposed model can inpaint 4096×4096 images in 87.3 milliseconds which is intractable with other learning-based methods due to limits of GPU memory.

4.4 Comparisons of CRA with Super-resolution

Figure 6, compares the high-resolution results of our CRA with those obtained via various super-resolution techniques. After obtaining a 512×512 inpainted result, we up-sample the output to the original size using different up-sampling methods including SRGAN [37], Nearest Neighbor, and Bicubic, then, we paste the filled contents to the original image. SRGAN [37] is a learning-based method that can perform $4\times$ super-resolution and the official pre-trained model was trained on DIV2K. We can observe from that the hole region generated by CRA is generally sharper and visually more consistent with surrounding areas.

4.5 Comparisons With Traditional Methods

Moreover, we compare our method with two commercial products based on PatchMatch [38] (Photoshop, Inpaint) and an open-source PatchMatch implementation for image inpainting [33] (Figure 7). We discover that PatchMatch-based methods are able to generate clear textures but with distorted structures incoherent with surrounding regions.

5 Conclusion

We presented a novel Contextual Residual Aggregated technique that enables more efficient and high-quality inpainting of ultra high-resolution images. Unlike other data-

driven methods, the increase of resolutions and hole size does not deteriorate the inpainting quality and does not considerably increase the processing time in our framework. When tested on high-resolution images between 1K and 2K, our model is extremely efficient where it is 3x$\sim$6x faster than the state-of-the-art on images of the same size. Also, it achieves better quality by reducing FID by 82% compared to the state-of-the-art. We also compared our method with commercial products that showed significant superiority in certain scenarios. So far, our method is the only learning-based technique that can enable end-to-end inpainting on the ultra-high-resolution image (4K$\sim$8K). In the future, we will explore similar mechanisms for other tasks such as image expansion, video inpainting and image blending.

Acknowledgment

We want to acknowledge Peng Deng, Shao Hua Chen, Xinjiang Sun, Chunhua Tian and other colleagues in Huawei Technologies for their support in the project.

References

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

[2] Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (2000) 417–424

[3] Criminisi, A., Pérez, P., Toyama, K.: Region filling and object removal by exemplar-based image inpainting. IEEE Transactions on image processing **13**(9) (2004) 1200–1212

[4] Drori, I., Cohen-Or, D., Yeshurun, H.: Fragment-based image completion. In: ACM Transactions on graphics (TOG). Volume 22.3., ACM (2003) 303–312

[5] He, K., Sun, J.: Statistics of patch offsets for image completion. In: European Conference on Computer Vision, Springer (2012) 16–29

[6] Wilczkowiak, M., Brostow, G.J., Tordoff, B., Cipolla, R.: Hole filling through photomontage. In: BMVC 2005-Proceedings of the British Machine Vision Conference 2005. (2005)

[7] Xu, Z., Sun, J.: Image inpainting by patch propagation using patch sparsity. IEEE transactions on image processing **19**(5) (2010) 1153–1165

[8] Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM (2001) 341–346

[9] Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: Proceedings of the seventh IEEE international conference on computer vision. Volume 2., IEEE (1999) 1033–1038

[10] Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (ToG) **36**(4) (2017) 107

[11] Liao, L., Hu, R., Xiao, J., Wang, Z.: Edge-aware context encoder for image inpainting. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE (2018) 3156–3160

[12] Oord, A.v.d., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759 (2016)

[13] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2536–2544
[14] Xiong, W., Yu, J., Lin, Z., Yang, J., Lu, X., Barnes, C., Luo, J.: Foreground-aware image inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 5840–5848
[15] Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., Li, H.: High-resolution image inpainting using multi-scale neural patch synthesis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 6721–6729
[16] Liu, G., Reda, F.A., Shih, K.J., Wang, T.C., Tao, A., Catanzaro, B.: Image inpainting for irregular holes using partial convolutions. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 85–100
[17] Song, Y., Yang, C., Lin, Z., Liu, X., Huang, Q., Li, H., Jay Kuo, C.C.: Contextual-based image inpainting: Infer, match, and translate. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 3–19
[18] Yan, Z., Li, X., Li, M., Zuo, W., Shan, S.: Shift-net: Image inpainting via deep feature rearrangement. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 1–17
[19] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Free-form image inpainting with gated convolution. arXiv preprint arXiv:1806.03589 (2018)
[20] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., Huang, T.S.: Generative image inpainting with contextual attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 5505–5514
[21] Zeng, Y., Fu, J., Chao, H., Guo, B.: Learning pyramid-context encoder network for high-quality image inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 1486–1494
[22] Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. IEEE Transactions on communications **31**(4) (1983) 532–540
[23] Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using laplacian pyramid of adversarial networks. In: Advances in neural information processing systems. (2015) 1486–1494
[24] Deshmukh, M., Bhosale, U.: Image fusion and image quality assessment of fused images. International Journal of Image Processing (IJIP) **4**(5) (2010) 484
[25] Sharifi, M., Fathy, M., Mahmoudi, M.T.: A classified and comparative study of edge detection algorithms. In: Proceedings. International conference on information technology: Coding and computing, IEEE (2002) 117–120
[26] Toet, A.: Image fusion by a ratio of low-pass pyramid. Pattern Recognition Letters **9**(4) (1989) 245–253
[27] Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
[28] Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015)
[29] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. In: Advances in neural information processing systems. (2017) 5767–5777
[30] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence **40**(6) (2017) 1452–1464
[31] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)

[32] Timofte, R., Gu, S., Wu, J., Van Gool, L., Zhang, L., Yang, M.H., Haris, M., et al.: Ntire 2018 challenge on single image super-resolution: Methods and results. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. (June 2018)
[33] Wen-Fu Lee, Yuan-Ting Hsieh, Z.F.E.: mage completion using patchmatch algorithm. https://github.com/YuanTingHsieh/Image_Completion Accessed: 2019-10-26.
[34] Wang, Z., Simoncelli, E.P., Bovik, A.C.: Multiscale structural similarity for image quality assessment. In: The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. Volume 2., Ieee (2003) 1398–1402
[35] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. In: Advances in neural information processing systems. (2016) 2234–2242
[36] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems. (2017) 6626–6637
[37] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 4681–4690
[38] Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. In: ACM Transactions on Graphics (ToG). Volume 28., ACM (2009) 24

Appendix

Network Architectures

In addition to Section 3.3 and Figure 2 in the main paper, we report more details of our network architectures. For simplicity, we denote them with K (kernel size), S (stride size), C (channel number) and D (dilation rate). D is neglected when D=1.

Coarse Network: downsample(2×) - K5S2C32 - K3S1C32 - K3S2C64 - K3S1C64 - K3S1C64 - K3S1C64 - K3S1C64 - K3S1C64 - K3S1C64 - K3S1C64D2 - K3S1C64D2 - K3S1C64D2 - K3S1C64D2 - K3S1C64D2 - K3S1C64D4 - K3S1C64D4 - K3S1C64D4 - K3S1C64D4 - K3S1C64D8 - K3S1C64D8 - K3S1C64 - K3S1C64 - K3S1C64 - upsample(2×) - K3S1C32 - upsample(2×) - K3S1C3 - clip - upsample(2×)

Refine Network: K5S2C32 - K3S1C32[$P^{l=1}$] - K3S2C64 - K3S1C64[$P^{l=2}$] - K3S2C128 - K3S1C128 - K3S1C128 - K3S1C128D2 - K3S1C128D4 - K3S1C128D8 - K3S1C128D16[$P^{l=$ - concat - K3S1C128 - upsample(2×) - K3S1C64 - K3S1C64 - concat - upsample(2×) - K3S1C32 - K3S1C32 - concat - upsample(2×) - K3S1C3 - clip

Attention Computing Branch: [$P^{l=3}$] - downsample (2×) - [P] - ACM - ATM

Attention Transfer Branch ($P^{l=3}$): [$P^{l=3}$] - ATM - K3S1C128 - concat

Attention Transfer Branch ($P^{l=2}$): [$P^{l=2}$] - ATM - K3S1C64 - K3S1C64D2 - concat

Table 3: Sources of some HD images used for test

Figure ID in the main paper	Image Source
Figure 3 top	http://www.sohu.com/a/117062677_189010
Figure 6 top	http://ow.ly/u8Wff
Figure 6 bottom	https://www.mafengwo.cn/yj/14103/s-0-0-0-0-1-0.html
Figure 1 topright	https://www.champaignoutdoors.com/kilimanjaro
Images in demo.pps	http://www.imecchina.com/news/1293274.html
	http://www.zdqx.com/wall/57962_6.html
	https://www.xuehua.us/2018/06/03/%E5%92%8C%E9%AB%98%E5%B0%94%E5%A4
	%AB%E5%98%89%E6%97%85%E4%B8%80%E9%81%93-%E6%8E%A2%E5%AF%BB%E4%BB
	%99%E6%B9%96%E8%BE%B9%E7%9A%84%E6%85%A2%E7%94%9F%E6%B4%BB/zh-tw/
	https://you.autohome.com.cn/details/68005/727cc0cec7214dd62e92d8f009e7adf9
	https://www.reyfoto.com/

Attention Transfer Branch ($P^{l=1}$): [$P^{l=1}$] - ATM - K3S1C32 - K3S1C32D2 - concat

Discriminator: K3S2C64 - K3S2C128 - K3S2C256 - K3S2C256 - K3S2C256 - K3S2C256 - fully connected to 1.

More Test Results on Places2

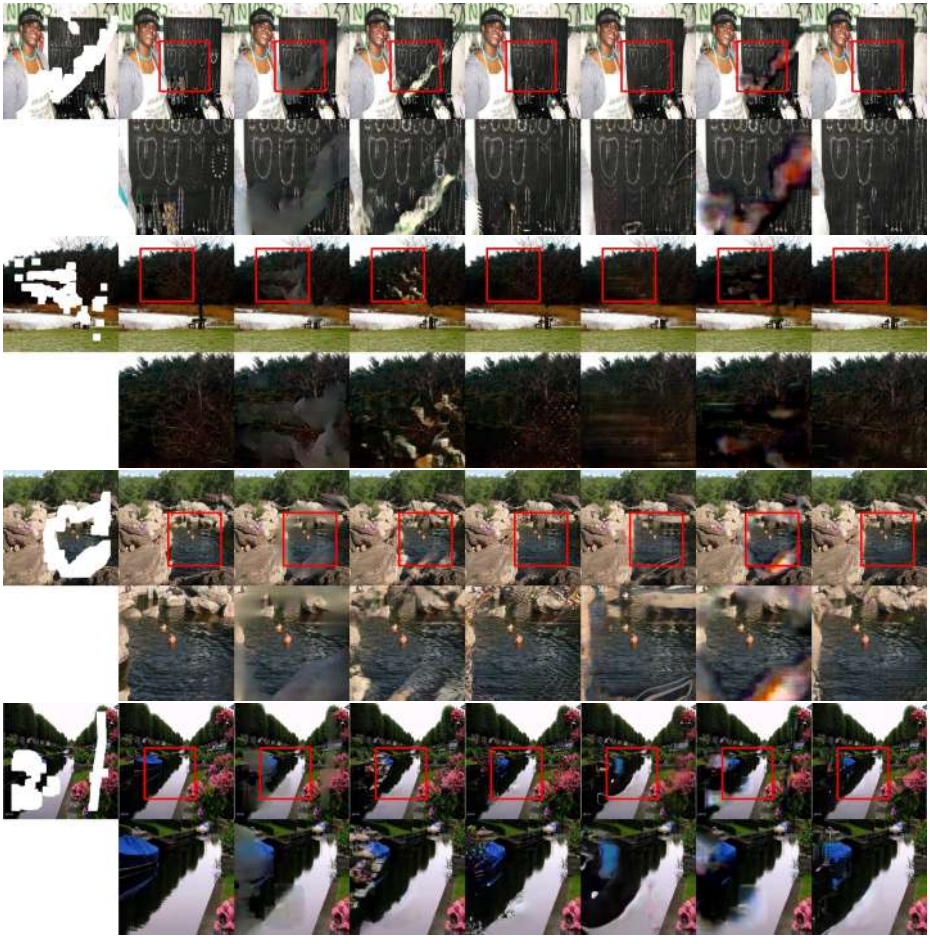
More test results on places2 are presented in Figures 8, 9, 10, with input size 512×512, 1024×1024, 2048×2048 respectively.

Sources of High-Resolution Images

Sources of the HD images in the main paper that are crawled from the internet are presented in Table 3.

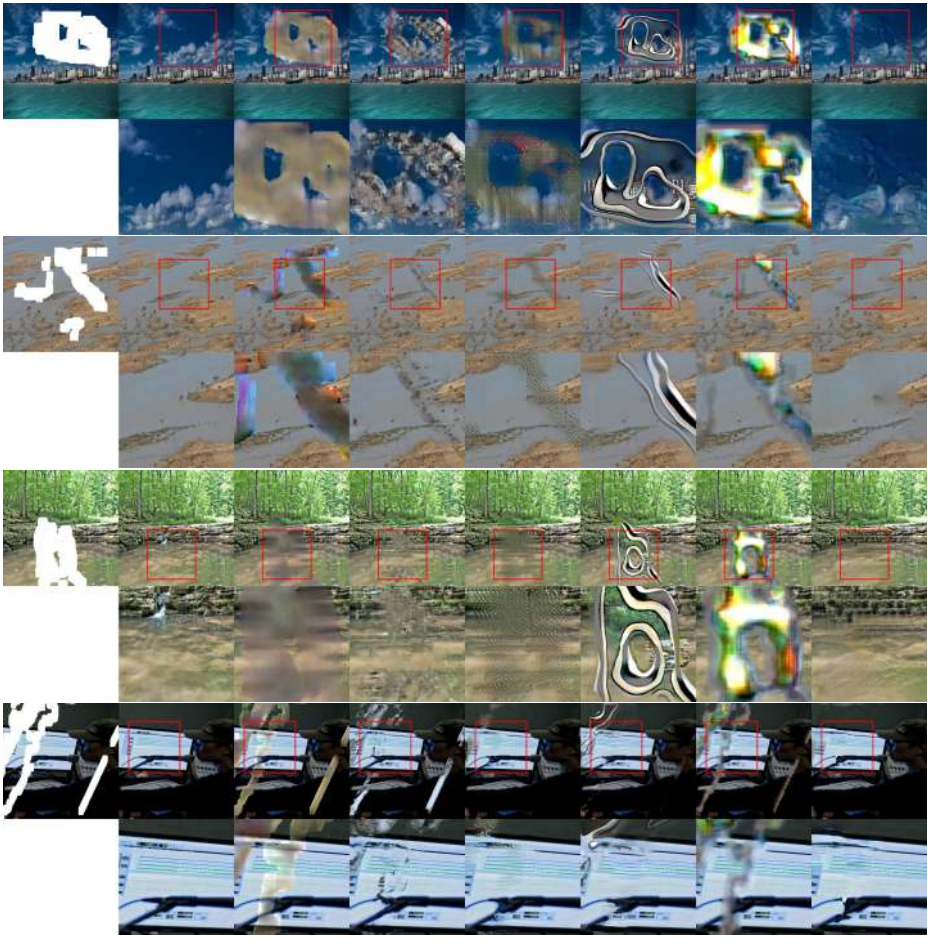
Failure Examples & Limitation

Some failure examples of our model are presented in Figure 11. Our model is prone to fail when the majority parts of a background object are missing (Referring to the bicycle and dog face in Figure. 4).



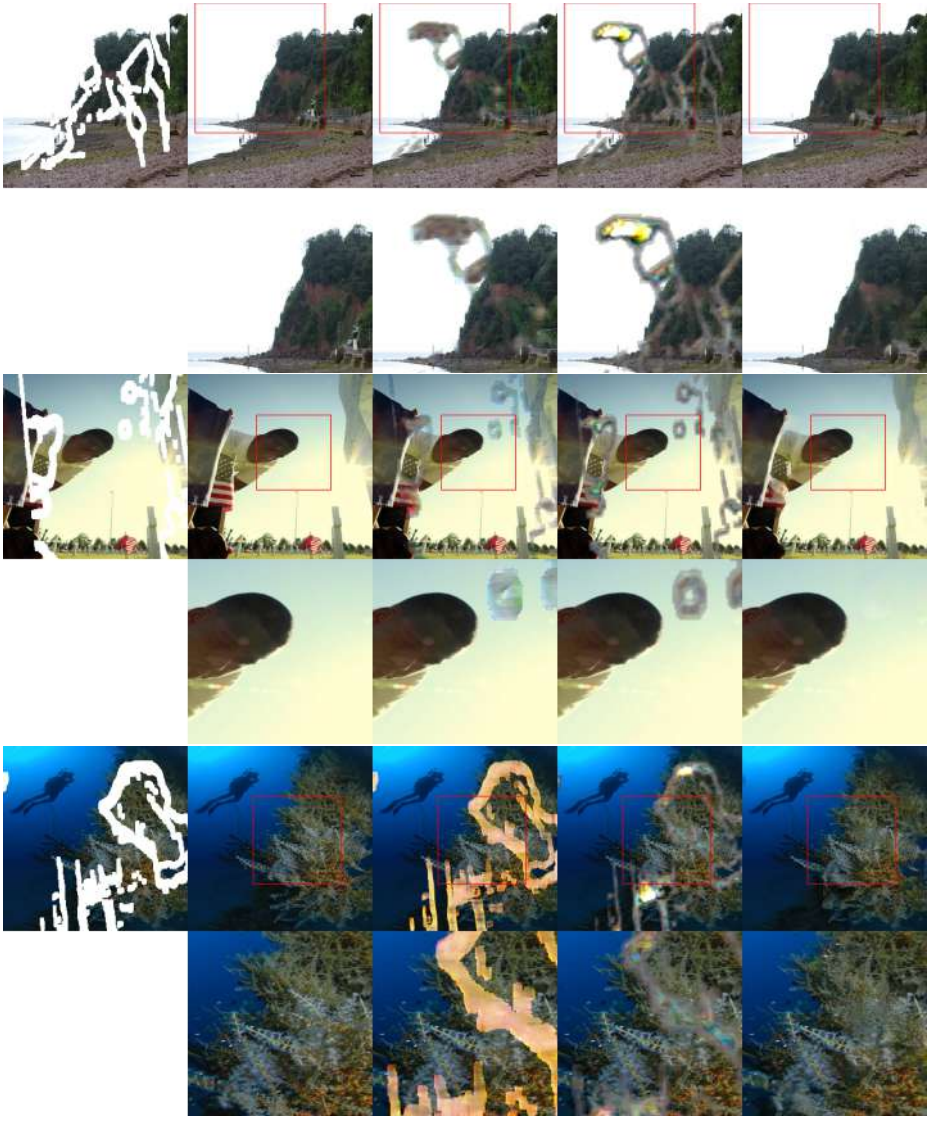
(a) Input (b) GT (c) global-local (d) DeepFillV1 (e) DeepFillV2 (f) PEN-Net (g) pconv (h) Ours

Fig. 8: Test results on places2 validation datasets with input size of 512 $\times$ 512.



(a) Input (b) GT (c) global-local (d) DeepFillV1 (e) DeepFillV2 (f) PEN-Net (g) pconv (h) Ours

Fig. 9: Test results on places2 validation datasets with input size of 1024 $\times$ 1024.



(a) Input (b) GT (c) global-local (d) pconv (e) Ours

Fig. 10: Test results on places2 validation datasets with input size of 2048 $\times$ 2048.

(a) Original (b) Input (c) Output (d) Original (e) Input (f) Output

Fig. 11: Failure examples of our model.