Generator pyramid for high-resolution image inpainting

Leilei Cao\textsuperscript{1,2} · Tong Yang\textsuperscript{2} · Yixu Wang\textsuperscript{2} · Bo Yan\textsuperscript{2} · Yandong Guo\textsuperscript{2}

Received: 29 December 2022 / Accepted: 17 April 2023 / Published online: 3 May 2023
© The Author(s) 2023

Abstract
Inpainting high-resolution images with large holes challenges existing deep learning-based image inpainting methods. We present a novel framework—PyramidFill for high-resolution image inpainting, which explicitly disentangles the task into two sub-tasks: content completion and texture synthesis. PyramidFill attempts to complete the content of unknown regions in a lower-resolution image, and synthesize the textures of unknown regions in a higher-resolution image, progressively. Thus, our model consists of a pyramid of fully convolutional GANs, wherein the content GAN is responsible for completing contents in the lowest-resolution masked image, and each texture GAN is responsible for synthesizing textures in a higher-resolution image. Since completing contents and synthesizing textures demand different abilities from generators, we customize different architectures for the content GAN and texture GAN. Experiments on multiple datasets including CelebA-HQ, Places2 and a new natural scenery dataset (NSHQ) with different resolutions demonstrate that PyramidFill generates higher-quality inpainting results than the state-of-the-art methods.

Introduction
Image inpainting, a fundamental low-level vision task, has attracted much attention from academia and industry. A wide range of vision and graphic applications refer to image inpainting, e.g., object removal \cite{1,2}, image restoration \cite{4}, manipulation \cite{24}, and super-resolution \cite{37}. Image inpainting aims to synthesize alternative contents in the missing regions of an image, which is visually realistic and semantically correct \cite{40}.

The existing methods for image inpainting generally fall into two categories: (1) Copying from low-level image features. (2) Generating new contents from high-level semantic features. The former ones attempt to fill holes by matching and copying patches from known regions \cite{1} or external database images \cite{7}. These approaches are effective for easy cases, e.g., uniform textured background inpainting. However, they cannot solve challenge cases where missing regions involve complex textures or nonrepetitive structures, due to the lack of high-level semantic understanding \cite{40}.

Since of the great development of advanced deep learning techniques \cite{6,13,33–35}, the approaches of generating new contents learn to capture the context representation of known regions and synthesize the missing contents by using deep Convolutional Neural Networks (CNN) in an end-to-end training manner. Context Encoder \cite{25} is the first CNN-based method to solve the image inpainting task, wherein a deep convolution encoder–decoder architecture is proposed and an adversarial loss \cite{5} is added to generate visually realistic results. The encoder–decoder architecture has been extended by many CNN-based inpainting models to get more reasonable and fine-detailed contents \cite{8,10,20,39–41,44,45,47}.

However, the end-to-end training encoder–decoder architecture still remains challenging to fill large holes in high-resolution images. As shown in Fig. 1, the results of PConv \cite{20} are obtained from its online demo,\textsuperscript{1} which illustrates that PConv fails to generate reasonable contents for these two cases, due to a lack of the guidance for filling the holes. Because some regions in holes are far away from the surrounding known regions, which causes the generator to produce semantically ambiguous contents or visually artifacts for the holes \cite{16}. In addition, high-resolution images make the discriminator easier to tell the fully recovered images apart from ground truth, thus drastically amplifying the gradient problem \cite{12,18}. An alternative way is to use two encoder–decoder architectures to separately fill contents

\textsuperscript{1} Leilei Cao did this work when he was with OPPO Research.

\textsuperscript{1} Leilei Cao
caoleilei@nwpu.edu.cn

\textsuperscript{1} School of Software, Northwestern Polytechnical University, Xi’an, China

\textsuperscript{2} OPPO Research, Shanghai, China

\textsuperscript{1} https://www.nvidia.com/research/inpainting.
and textures for the unknown regions [40]. Even so, only a few methods can process images with resolution smaller than $512 \times 512$, wherein the masked area is generally less than 25% of the whole image [19, 41].

We observe that filling the contents and textures demands quite different abilities from the generator. Content completion relies more on capturing the high-level semantics and the global structure of the image, yet low-level features and local texture statistics of the image is more critical for the texture synthesis. Furthermore, content completion can be regarded as an image generation task [5], and texture synthesis can be treated as an image to image translation task [9].

Motivated by the observation that high-resolution image inpainting could be disentangled into content completion and texture synthesis, we propose a simple yet powerful high-resolution image inpainting framework named PyramidFill, which attempts to fill large holes in images with high resolution reaching to $1024 \times 1024$. Our key insight is that we can fill the contents for the easier low-resolution images and synthesize the textures for the high-resolution progressively, wherein the high-resolution image is formed into a pyramid of different scale images by down-sampling. PyramidFill consists of a pyramid of fully convolutional Generative Adversarial Networks (GANs), wherein the content GAN is responsible for generating contents in the lowest-resolution masked image in image pyramid, and each texture GAN is responsible for synthesizing textures in a high-resolution image. Unlike the previous inpainting methods, e.g., DeepFill V2 [41] and PEN-Net [42], which complete contents and synthesize textures simultaneously with an encoder–decoder architecture or a pyramid-context encoder, our proposed method fill contents for the masked image before synthesizing textures with different GANs. Under this manner, each GAN can focus on their respective sub-tasks to acquire better results.

Our major contributions can be summarized as follows:

- We propose a novel framework for high-resolution image inpainting, termed PyramidFill. Our framework disentangles the task of high-resolution image inpainting into two sub-tasks: low-resolution content completion and high-resolution texture synthesis.
- We design a content GAN architecture for filling contents in the low-resolution masked image, and a texture GAN architecture for synthesizing textures in a high-resolution image progressively.
- Extensive experiments on CelebA-HQ and Place2 show that PyramidFill outperforms the previous methods on these benchmarks.
- We introduce a new dataset of natural scenery with high resolution $1920 \times 1080$ for real-image inpainting applications.

Related work

Deep image inpainting

A variety of CNN-based approaches have been proposed for image inpainting. Pathak et al. [25] first introduced an encoder–decoder architecture for the image inpainting task, as well as a pixel-wise reconstruction loss and an adversarial loss. Based on Pathak’s work, Iizuka et al. [8] proposed a fully convolutional GAN model with an extra local discriminator to ensure local image coherency. Yang et al. [36] proposed a multi-scale neural patch synthesis approach based on joint optimization of image content and texture constraints. Based on two-stage encoder–decoder architectures, observing ineffectiveness of CNN in modeling long-term correlations between distant contextual information and the hole regions, Yu et al. [40] presented a novel contextual attention layer to integrate in the second stage, which used the features of known patches as convolutional filters to process the generated patches. The aforementioned approaches were based on vanilla convolutions that treated all input pixels as same valid ones, which is not reasonable for masked holes. To address this limitation, Liu et al. [20] proposed a partial convolutional layer for irregular holes in image inpainting, comprising a masked and re-normalized convolution operation followed by a mask-update step. Partial Convolutions could be viewed as hard-mask convolutional layers, Yu et al. [41] proposed gated convolution to learn a dynamic feature gating mechanism for each channel and each spatial location across all layers, which could be viewed as a soft-mask convolutional layer. Yang et al. [38] proposed a multi-task learning framework to incorporate the image structure knowledge to assist image inpainting, which trained a shared generator to simultaneously complete the masked image and corresponding structures—edge and gradient. Liu et al. [21] proposed a mutual encoder–decoder CNN for joint recovering struc-
tured image pyramid representations, Shocher et al. \[28\] proposed a contextual residual aggregation mechanism that could produce high-frequency residuals for missing contents by weighted aggregating residuals from contextual patches, thus only requiring a low-resolution prediction from the network. Zeng et al. \[43\] presented a guided upsampling network to generate high-resolution image inpainting results, by extending the contextual attention module which borrowed high-resolution feature patches in the input image.

**High-resolution image inpainting**

Most recently, a few works have been presented for high-resolution image inpainting. Instead of directly filling holes in high-resolution images, Yi et al. \[39\] proposed a Laplacian pyramid framework to generate images in a coarse-to-fine fashion. At each level of the pyramid, a separate generation model was trained using the GAN approach. Shaham et al. \[27\] introduced SinGAN to learn from a single natural image, which contained a pyramid of CNN features. SinGAN was trained using the GAN approach. The filling of a high-resolution image starts with the lowest-scale image \(x_0\), wherein \(\{G_0, D_0\}\) are trained to complete the contents, and then progressively synthesize the finer textures at the higher-scale image by training \(\{G_i, D_i\}\), \(i = 1, 2, 3, 4\). The generator \(G_0\) takes concatenation of \(z_0\) and \(m_0\) as input, and output the predicted image \(x_0\) with the same size as input. We then replace the masked region of \(z_0\) using the predicted image to get the inpainting result \(y_0\).

\[
y_0 = z_0 \odot (1 - m_0) + G_0([z_0, m_0]) \odot m_0.
\]

After training the generator \(G_0\), the inpainting result \(y_0\) and input image \(z_1\) are given to the generator \(G_1\) to output the predicted image \(x_1\) with the same size as \(z_1\), and it replaces the masked region of \(z_1\) to get the inpainting result \(y_1\). Training the other PatchGANs is similar to training \(\{G_1, D_1\}\), given the corresponding input image \(z_i\) and the lower-scale inpainting result \(y_{i-1}\) to get the current-scale inpainting result \(y_i\), i.e.,

\[
y_i = z_i \odot (1 - m_i) + G_i(z_i, y_{i-1}) \odot m_i, \quad i = 1, 2, 3, 4.
\]

**Generators and discriminators**

Since completing contents and synthesizing textures demand quite different abilities from generators, we customize different architectures for \(\{G_0, D_0\}\) and \(\{G_i, D_i\}\), \(i = 1, 2, 3, 4\).

Figure 2 shows the architecture of a PatchGAN for completing contents on the lowest-scale input image \(z_0\). For the generator network, we maintain the spatial size of the feature maps since of low-scale input images, instead of the encoder–decoder architecture used in many methods. The input images are first passed through four gated convolutional layers \[41\], which are then split into two branches along the channel dimension. Four dilated gated convolutional layers are adopted in the upper branch to expand the size of receptive fields for exploring the global structures of input images, four gated convolutional layers are simultaneously adopted in the lower branch for exploiting fine contents. Feature maps from two branches are then concatenated to proceed the last four gated convolutional layers to predict the contents in the masked region. The discriminator network consists of eight convolutional layers, wherein no down-sampling operations are adopted to ensure the spatial size of the output as the same with the input image. It can better capture fine details in the lowest-scale image.

**Method**

In this section, we first introduce the pipeline of our proposed PyramidFill, and then present the generator and discriminator networks in each level, as well as the loss functions.

**Pipeline of PyramidFill**

Figure 2 illustrates the pipeline of PyramidFill, which consists of a pyramid of PatchGANs \[9, 15\]: \(\{G_i, D_i\}\), \(i = 0, 1, 2, 3, 4\). For training, given a high-resolution image \(x\), we sample a binary image mask \(m\) at a random location (e.g., center area in Fig. 2). Input image \(z\) is masked from the original image as \(z = x \odot (1 - m) + m\). The original image, input image, mask are separately downsampling to be the pyramids of images by a factor \(r^{i-1}\) (we choose \(r = 2\)): \(x_i, z_i, m_i\), \(i = 0, 1, 2, 3, 4\). Each Generator \(G_i\) is responsible for filling the corresponding-scale masked image \(z_i\), trained against the image \(x_i\), where \(G_i\) learns to fool the discriminator \(D_i\).
The architecture of a PatchGAN for synthesizing textures is presented in Fig. 4, and \{G_1, D_1\}, \{G_2, D_2\}, \{G_3, D_3\}, \{G_4, D_4\} share this same architecture yet progressively training. There are two stages in the generator, and we call the first stage as the super-resolution network, the second stage as the refinement network. The super-resolution network predicts a higher-resolution image from its lower-resolution counterpart which is taken from the inpainting result of the lower-scale generator. We use the predicted higher-resolution image to replace the masked region of the corresponding-scale input image, which then passes through the refinement network to output a finer result. Inspired by ESRGAN [31] and SRResNet [14], we design a simple yet powerful super-resolution network, wherein all vanilla convolutions in RRDB module [31] are replaced with gated convolutions, and we only adopt two RRDB modules. The upsampling operator uses the sub-pixel layer with the factor 2 to increase the resolution of the input image. In the refinement network, there are only six gated convolutional layers with a shortcut connection within the middle four layers. In the discriminator, we use four strided convolutions with stride 2 to reduce the spatial size of feature maps due to memory cost.

Loss function

To stabilize the training of PatchGANs, the spectral normalization [23] is applied in all discriminators. We use a hinge
loss [41] as objective function for all PatchGANs \( \{G_i, D_i\}, \)
\( i = 0, 1, 2, 3, 4 \):
\[
\begin{align*}
L_{D_i}^{adv} &= E_{x_i \sim p_{data}(x_i)}[\text{ReLU}(1 - D_i(x_i))] \\
&\quad + E_{z_i \sim p_z(z_i)}[\text{ReLU}(1 + D_i(G_i(z_i)))], \\
L_{G_i}^{adv} &= -E_{z_i \sim p_z(z_i)}[D_i(G_i(z_i))], \\
L_{G_i}^{style} &= \left\| G\phi(x_i, y_i) - G\phi(x_i) \right\|_1,
\end{align*}
\]
where \( x_i \) and \( z_i \) represent the real image and the masked input image, respectively. \( D_i \) and \( G_i \) denote the spectral-normalized discriminator and the generator, respectively.

Inspired by U-Net GAN [26], we use a per-pixel consistency regularization technique when training \( D_0 \), encouraging the discriminator to focus more on semantic and structural changes between real and fake images. This technique can further enhance the quality of the generated sample \( x_0 \). We train the discriminator to provide consistent per-pixel predictions, by introducing the consistency regularization loss term in the discriminator objective function:
\[
L_{D_0}^{cons} = \left\| D_0(y_0) - \left( D_0(x_0) \odot (1 - m_0) \\
+ D_0(x_0) \odot m_0 \right) \right\|^2,
\]
where \( \| \cdot \| \) denotes the \( L^2 \) norm. This consistency loss is taken between the per-pixel output of \( D_0 \) on the composing image \( y_0 \) and the composite of output of \( D_0 \) on real and fake images, penalizing the discriminator for inconsistent predictions.

In addition, a pixel reconstruction loss is introduced for training the generators, because the task of generators is not only fool the discriminators but also to generate image being similar to the real image.
\[
L_{x_i}^{re} = \frac{1}{N_{x_i}} \| x_i - \hat{G}(x_i) \|_1,
\]
where \( N_{x_i} \) is the number elements in the image \( x_i \), which is the predicted image by the generators.

We introduce additional losses for training the texture generators, perceptual loss [11] and style loss [10, 20]. The perceptual loss penalizes results that are not perceptually similar by computing \( L^1 \) distance between feature maps of a pre-trained network,
\[
L_{x_i}^{per} = \sum_q \frac{1}{N_q} \left\| \phi_q(x_i) - \phi_q(\hat{G}(x_i)) \right\|_1,
\]
where \( N_q \) indicates the number of elements in the \( q \)-th layer, and \( \phi_q \) is the feature map of the \( q \)-th layer extracted from the VGG-16 network pre-trained on ImageNet [29], and we choose the feature maps from layers \( pool1 \), \( pool2 \) and \( pool3 \). Style loss compares the content of two images by using Gram matrix. Given feature maps of sizes \( C_q \times H_q \times W_q \), style loss is computed as follows:
\[
L_{x_i}^{style} = \left\| G\phi(x_i) - G\phi(x_i) \right\|_1,
\]
where \( G\phi \) is a \( C_q \times C_q \) Gram matrix constructed from feature maps \( \phi_q \), which are the same with that used in the perceptual loss.

The overall loss functions for training \( \{G_0, D_0\} \) are shown as below:
\[
\begin{align*}
L_{D_0} &= L_{D_0}^{adv} + L_{D_0}^{cons}, \\
L_{G_0} &= L_{G_0}^{adv} + L_{x_i}^{re}. \\
\end{align*}
\]
And the whole loss functions for training \( \{G_i, D_i\}, \ i = 1, 2, 3, 4 \) are as follows:
\[
\begin{align*}
L_{D_i} &= L_{D_i}^{adv}, \\
L_{G_i} &= \lambda_a L_{G_i}^{adv} + \lambda_r L_{x_i}^{re} + \lambda_p L_{x_i}^{per} + \lambda_{si} L_{x_i}^{style} \\
&\quad + \lambda_{s} L_{x_i}^{re} + \lambda_p L_{x_i}^{per} + \lambda_{si} L_{x_i}^{style}. \\
\end{align*}
\]
For our experiments, we empirically set [20] \( \lambda_a = 0.001 \), \( \lambda_r = 0.1 \), \( \lambda_p = 0.1 \) for all generators, and \( \lambda_{si} = 1, \lambda_{s} = 50, \lambda_{s} = 120, \lambda_{s} = 250 \) for different generators, respectively.
Experiments

Datasets. We evaluate PyramidFill on three datasets: CelebA-HQ [12], Places2 [46], and our new collected NSHQ dataset. CelebA-HQ contains 30,000 high-quality images at 1024 × 1024 resolution focusing on human faces. For Places2, we randomly select 20 categories from the 365 categories to form a subset of 100,000 images. The NSHQ dataset includes 5000 high-quality natural scenery images at 1920 × 1080 or 1920 × 1280 resolution, collecting from the Internet. For Places2 and NSHQ, images are randomly cropped to 512 × 512 and 1024 × 1024, respectively. Therefore, there is no \{G_4, D_4\} when training on the Places2 subset. For all datasets, we randomly use 90% of images for training and 10% of images for testing.

Compared methods. We compare PyramidFill with five state-of-the-art approaches: Global & Local [8], DeepFill v1 [40], PConv [20], PEN-Net [42], DeepFill v2 [41].

Evaluation metrics. We perform evaluations using three evaluation metrics: (1) relative L1 loss (mean average error); (2) peak signal-to-noise ratio (PSNR); (3) structural similarity (SSIM) [32]. These metrics are used to compare the low-level differences between the recovered image and the ground truth.

Implementation details. We implement our method in PyTorch. We train the model by the Adam optimizer with \(\beta_1 = 0.5\) and \(\beta_2 = 0.999\), and the learning rate is set to 0.0002 and 0.0008 for the generators and discriminators, respectively. The content GAN (\(\{G_0, D_0\}\)) and textures GANs (\(\{G_i, D_i\}, i = 1, 2, 3, 4\)) are trained separately, in which the content GAN is trained firstly and the texture GANs are trained sequentially. All networks are trained on 8 NVIDIA V100 GPUs (32 G) with batch size of 8. When training on CelebA-HQ, we use regular masks and random free-form masks [41], wherein the regular masks cover image center with half of image size. Free-form masks are generated automatically by drawing lines and rotating angles during training, and the details can refer to [41]. For Places2 and NSHQ, random free-form masks [41] are employed for training, while irregular masks from PConv [20] are used for testing.

Quantitative evaluation

We conduct quantitative comparisons on CelebA-HQ dataset and Places2 subset. For a fair evaluation, all input images are resized to 256 × 256 and 512 × 512, respectively. Because a few compared official pre-trained models are only trained on 256 × 256 images. Tables 1 and 2 report the quantitative comparisons on CelebA-HQ testing set with 256 × 256 images and 512 × 512 images, respectively. The images are masked with center holes. It shows that our PyramidFill outperforms existing state-of-the-art methods with significant advantages. Tables 3 and 4 present the quantitative comparisons on Places2 testing set with 256 × 256 images and 512 × 512 images masked with irregular masks, respectively. The results are categorized according to the ratios of the hole regions versus the image size. It shows that our PyramidFill performs much better than the state-of-the-art methods, especially for filling large holes in images. We also compare the computational complexity and uptime between our PyramidFill and some SOTA methods. Table 5 shows that PConv has the fewest parameters and FLOPs, which needs 8 ms for processing a 512 × 512 image on a single NVIDIA RTX 3090 GPU. DeepFill v2 and our model increase the parameters, which require much more inference time.

Qualitative evaluation

The qualitative comparisons on three datasets are shown in Fig. 5. It shows that Global & Local and DeepFill v1 often generate heavy artifacts, even the holes are not very large. PConv, PEN-Net and DeepFill v2 can generate correct contents for completing faces, yet lacking of detailed textures. By contrast, our model PyramidFill can generate more realistic results on the corrupted faces with finer textures. When dealing with the corrupted images from Places2 and NSHQ, PConv and PEN-Net also produce obvious artifacts. DeepFill v2 and our PyramidFill both can generate reasonable contents and detailed textures. Results obtained from our PyramidFill
Table 3  Quantitative comparisons on Places2 subset with input size 256 × 256 resolution, where the inputs are with irregular holes

| Mask       | Methods       | L1 Loss ↓ | PSNR ↑ | SSIM ↑ |
|------------|---------------|-----------|--------|--------|
| 0–10%      | Global & Local| 0.0109    | 29.43  | 0.9443 |
|            | DeepFill v1   | 0.0101    | 31.51  | 0.9571 |
|            | PConv         | 0.0091    | 32.27  | 0.9487 |
|            | PEN-Net       | 0.0079    | 32.66  | 0.9622 |
|            | DeepFill v2   | 0.0069    | 35.77  | 0.9727 |
|            | Ours          | 0.0016    | 38.30  | 0.9864 |
| 10–20%     | Global & Local| 0.0231    | 24.90  | 0.8724 |
|            | DeepFill v1   | 0.0225    | 25.44  | 0.8928 |
|            | PConv         | 0.0172    | 28.13  | 0.9027 |
|            | PEN-Net       | 0.0190    | 26.84  | 0.9017 |
|            | DeepFill v2   | 0.0136    | 30.22  | 0.9111 |
|            | Ours          | 0.0091    | 31.86  | 0.9162 |
| 20–30%     | Global & Local| 0.0372    | 22.42  | 0.7963 |
|            | DeepFill v1   | 0.0384    | 22.25  | 0.8179 |
|            | PConv         | 0.0265    | 25.68  | 0.8334 |
|            | PEN-Net       | 0.0331    | 23.71  | 0.8287 |
|            | DeepFill v2   | 0.0227    | 26.85  | 0.8768 |
|            | Ours          | 0.0129    | 29.12  | 0.8569 |
| 30–40%     | Global & Local| 0.0514    | 20.73  | 0.7265 |
|            | DeepFill v1   | 0.0544    | 20.27  | 0.7471 |
|            | PConv         | 0.0360    | 23.96  | 0.7765 |
|            | PEN-Net       | 0.0479    | 21.66  | 0.7588 |
|            | DeepFill v2   | 0.0325    | 24.62  | 0.8213 |
|            | Ours          | 0.0132    | 30.06  | 0.8499 |
| 40–50%     | Global & Local| 0.0665    | 19.40  | 0.6559 |
|            | DeepFill v1   | 0.0706    | 18.90  | 0.6757 |
|            | PConv         | 0.0475    | 22.40  | 0.7121 |
|            | PEN-Net       | 0.0656    | 19.94  | 0.6839 |
|            | DeepFill v2   | 0.0441    | 22.79  | 0.7596 |
|            | Ours          | 0.0246    | 26.19  | 0.7680 |

The best results are in bold

Real applications

We study real use cases of image inpainting on high-resolution images using our PyramidFill, e.g., freckles removal, face editing, watermark removal, and general object removal in natural scenery. In the first row of Fig. 6, our model successfully removes the freckles on the face, as well as changing eyes. In the second row, the watermarks on the original image are removed successfully. In the third row, we remove the giraffes on the grassland and recover the background. In the last row, a boat is removed from the river. All inpainting results are realistic and with fine-grained details.

Table 4  Quantitative comparisons on Places2 subset with input size 512 × 512 resolution, where the inputs are with irregular holes

| Mask       | Methods       | L1 Loss ↓ | PSNR ↑ | SSIM ↑ |
|------------|---------------|-----------|--------|--------|
| 0–10%      | PConv         | 0.0068    | 34.05  | 0.9647 |
|            | DeepFill v2   | 0.0070    | 34.82  | 0.9710 |
|            | Ours          | 0.0012    | 37.93  | 0.9869 |
| 10–20%     | PConv         | 0.0141    | 29.11  | 0.9168 |
|            | DeepFill v2   | 0.0146    | 29.14  | 0.9272 |
|            | Ours          | 0.0075    | 31.19  | 0.9263 |
| 20–30%     | PConv         | 0.0238    | 26.09  | 0.8584 |
|            | DeepFill v2   | 0.0248    | 25.80  | 0.8716 |
|            | Ours          | 0.0132    | 28.09  | 0.8460 |
| 30–40%     | PConv         | 0.0343    | 23.96  | 0.8006 |
|            | DeepFill v2   | 0.0343    | 23.59  | 0.8156 |
|            | Ours          | 0.0131    | 28.92  | 0.8433 |
| 40–50%     | PConv         | 0.0472    | 22.09  | 0.7360 |
|            | DeepFill v2   | 0.0487    | 21.79  | 0.7548 |
|            | Ours          | 0.0241    | 25.47  | 0.7549 |

The best results are in bold

Table 5  Comparisons on the computational complexity and uptime with input size 512 × 512 resolution

| Methods       | params | FLOPs | Uptime |
|---------------|--------|-------|--------|
| PConv         | 25.78M | 75.8G | 8 ms   |
| DeepFill v2   | 26.95M | 724G  | 79 ms  |
| Ours          | 32.86M | 846G  | 138 ms |

All models are performed on a single NVIDIA RTX 3090 GPU

Furthermore, our PyramidFill can also be used for more real applications, e.g., removing wrinkles on faces, changing hairstyles, restoring old photos.

Ablation study

Consistency regularization loss To demonstrate the effects of consistency regularization loss used in the discriminator $D_0$, we retrain $\{G_0, D_0\}$ yet without consistency regularization loss on CelebA-HQ dataset masked with center holes. The results are provided in Table 6, which shows that the consistency regularization loss can definitely improve the performance of PyramidFill on completing contents.

Refinement network In Texture Generators, a refinement network is designed as a second-stage for generating a finer result. We provide ablation experiments on CelebA-HQ dataset with 128 × 128 images masked with center holes. For fair comparisons, two-stage networks are merged into a one-stage network, wherein the feature maps of the upsampling operator in the super-resolution network are directly inputted to the refinement network, and there is no coarse

The best results are in bold
result to output. As shown in Fig. 7, the two-stage network can generate a more photo-realistic inpainting result.

**Discussion**

Zooming in Fig. 6, we find that our model works not so well for synthesizing textures of $1024 \times 1024$ images, which is fidelity defect. This illustrates the limitation of our proposed Generator pyramids when processing $1024 \times 1024$ images. Analyzing the architecture of texture Generator $G_i, i = 1, 2, 3, 4$, we find that this architecture is not available to model long-range dependencies in the whole image, which results in lacking of global visual field. This limitation may be improved by adopting the architecture of visual Transformers [17, 30] to design the generator or using the diffusion model [22] to synthesizing the textures for high-resolution images. This might be an interesting topic for our future work.

**Conclusions**

We present PyramidFill, a novel framework for the high-resolution image inpainting task. PyramidFill consists of a pyramid of PatchGANs, wherein the content GAN is responsible for generating contents in the lowest-resolution corrupted images, and each texture GAN is responsible for synthesizing textures for higher-resolution images progressively. We customized the generators and discriminators for the content GAN and texture GAN, respectively. Our model is trained on several datasets to evaluate its ability to fill correct contents and realistic textures for high-resolution image inpainting. Quantitative and qualitative results demonstrate
The superiority of PyramidFill, comparing with several state-of-the-art methods.

Declarations

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

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Barnes C, Shechtman E, Finkelstein A, Goldman DB (2009) PatchMatch: a randomized correspondence algorithm for structural image editing. ACM Trans Graph 28(3):24
2. Criminisi A, Perez P, Toyama K (2003) Object removal by exemplar-based inpainting. In: CVPR
3. Denton E, Chintala S, Szlam A, Fergus RD (2015) Generative image models using a Laplacian pyramid of adversarial networks. In: NIPS
4. Du W, Hu C, Yang H (2020) Learning invariant representation for unsupervised image restoration. In: CVPR
5. Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: NIPS
6. Guo Y, Liu Y, Oerlemans A, Lao S, Wu S, Lew MS (2016) Deep learning for visual understanding: a review. Neurocomputing 187:27–48
7. Hays J, Efros AA (2008) Scene completion using millions of photographs. Commun ACM 51:87–94
8. Iizuka S, Simo-Serra E, Ishikawa H (2017) Globally and locally consistent image completion. ACM Trans Graph 36(4):107
9. Isola P, Zhu J-Y, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. In: CVPR
10. Jo Y, Park J (2019) SC-FEGAN: face editing generative adversarial network with user’s sketch and color. In: ICCV
11. Justin J, Alexandre A, Li F-F (2016) Perceptual losses for real-time style transfer and super-resolution. In: ECCV
12. Karras T, Aila T, Laine S, Lehtinen J (2018) Progressive growing of GANs for improved quality, stability, and variation. In: ICLR
13. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
14. Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, Shi W (2017) Photo-realistic single image super-resolution using a generative adversarial network. In: CVPR
15. Li C, Wand M (2016) Precomputed real-time texture synthesis with Markovian generative adversarial networks. In: ECCV
16. Li J, Wang N, Zhang L, Du B, Tao D (2020) Recurrent feature reasoning for image inpainting. In: CVPR
17. Li W, Lin Z, Zhou K, Qi L, Wang Y, Jia J (2022) MAT: mask-aware transformer for large hole image inpainting. In: CVPR, pp 10748–10758
18. Li X, Guo Q, Lin D, Li P, Feng W, Wang S (2022) MISF: multi-level interactive Siamese filtering for high-fidelity image inpainting. In: CVPR, pp 1859–1868
19. Liao L, Xiao J, Wang Z, Lin C-W, Satoh S (2021) Image inpainting guided by coherence priors of semantics and textures. In: CVPR, pp 6535–6544
20. Liu G, Reda FA, Shih KJ, Wang T-C, Tao A, Catanazzo B (2018) Image inpainting for irregular holes using partial convolutions. In: ECCV
21. Liu H, Jiang B, Song Y, Huang W, Yang C (2020) Rethinking image inpainting via a mutual encoder-decoder with feature equalizations. In: ECCV
22. Lugmayr A, Danelljan M, Romero A, Yu F, Timofte R, Van Gool L (2022) RePaint: inpainting using denoising diffusion probabilistic models. In: CVPR, pp 11451–11461
23. Miyato T, Kataoka T, Koyama M, Yoshida Y (2018) Spectral normalization for generative adversarial networks. In: ICLR
24. Pan X, Zhan X, Dai B, Lin D, Loy CC, Luo P (2020) Exploiting deep generative prior for versatile image restoration and manipulation. In: ECCV
25. Patankar D, Krähenbühl P, Donahue J, Darrell T, Efros AA (2016) Context encoders: feature learning by inpainting. In: CVPR
26. Schönfeld E, Schiele B, Khoreva A (2020) A U-Net based discriminator for generative adversarial networks. In: CVPR
27. Shaham TR, Dekel T, Michaeli T (2019) SinGAN: learning a generative model from a single natural image. In: ICCV
28. Shoemaker A, Gandelman Y, Mosseri I, Yarom M, Irani M, Freeman WT, Dekel T (2020) Semantic pyramid for image generation. In: CVPR
29. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR
30. Wan Z, Zhang J, Chen D, Liao J (2021) High-fidelity pluralistic image completion with transformers. In: ICCV
31. Wang X, Yu K, Wu S, Gu J, Liu Y, Dong C, Qiao Y, Change Loy C (2018) ESRGAN: enhanced super-resolution generative adversarial networks. In: ECCVW
32. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
33. Xiao Z, Zhang H, Tong H, Xu X (2022) An efficient temporal network with dual self-distillation for electroencephalography signal classification. In: BIBM, pp 1759–1762
34. Xing H, Xiao Z, Qu R, Zhu Z, Zhao B (2022) An efficient federated distillation learning system for multitask time series classification. IEEE Trans Instrum Meas 71:1–12
35. Xiao Z, Xu X, Xing H, Song F, Wang X, Zhao B (2021) A federated learning system with enhanced feature extraction for human activity recognition. Knowl-Based Syst 229:107338
36. Yang C, Lu X, Lin Z, Shechtman E, Wang O, Li H (2017) High-resolution image inpainting using multi-scale neural patch synthesis. In: CVPR
37. Yang F, Yang H, Fu J, Lu H, Guo B (2020) Learning texture transformer network for image super-resolution. In: CVPR
38. Yang J, Qi Z, Shi Y (2020) Learning to incorporate structure knowledge for image inpainting. In: AAAI
39. Yi Z, Tang Q, Azizi S, Jang D, Xu Z (2020) Contextual residual aggregation for ultra high-resolution image inpainting. In: CVPR
40. Yu J, Lin Z, Yang J, Shen X, Lu X, Huang TS (2018) Generative image inpainting with contextual attention. In: CVPR
41. Yu J, Lin Z, Yang J, Shen X, Lu X, Huang TS (2019) Free-form image inpainting with gated convolution. In: CVPR
42. Zeng Y, Fu J, Chao H, Guo B (2019) Learning pyramid-context encoder network for high-quality image inpainting. In: CVPR
43. Zeng Y, Lin Z, Yang J, Zhang J, Shechtman E, Lu H (2020) High-resolution image inpainting with iterative confidence feedback and guided upsampling. In: ECCV
44. Zeng Y, Lin Z, Lu H, Patel VM (2021) CR-Fill: generative image inpainting with auxiliary contextual reconstruction. In: ICCV, pp 14144–14153
45. Zheng C, Cham T-J, Cai J, Phung D (2022) Bridging global context interactions for high-fidelity image completion. In: CVPR, pp 11512–11522
46. Zhou B, Lapedriza A, Khosla A, Oliva A, Torralba A (2017) Places: a 10 million image database for scene recognition. IEEE Trans Pattern Anal Mach Intell 40(6):1452–1464
47. Zhu M, He D, Li X, Li C, Li F, Liu X, Ding E, Zhang Z (2021) Image inpainting by end-to-end cascaded refinement with mask awareness. IEEE Trans Image Process 30:4855–4866

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.