Pattern Recognition 134 (2023) 109046

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Deep learning for image inpainting: A survey

Hanyu Xiang[a,b], Qin Zou[b,*], Muhammad Ali Nawaz[b], Xianfeng Huang[a,c], Fan Zhang[a], Hongkai Yu[d]

[a] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, China
[b] Institute of Artificial Intelligence, School of Computer Science, Wuhan University, China
[c] Institute of Yangtze River Civilization Archaeology Research, Wuhan University, China
[d] Department of Electrical Engineering and Computer Science, Cleveland State University, USA

ARTICLE INFO

Article history:
Received 22 December 2021
Revised 16 July 2022
Accepted 15 September 2022
Available online 20 September 2022

Keywords:
Image inpainting
Image restoration
Generative adversarial network
Convolutional neural network

ABSTRACT

Image inpainting has been widely exploited in the field of computer vision and image processing. The main purpose of image inpainting is to produce visually plausible structure and texture for the missing regions of damaged images. In the past decade, the success of deep learning has brought new opportunities to many vision tasks, which promoted the development of a large number of deep learning-based image inpainting methods. Although these methods have many similarities, they also have their own characteristics due to the differences in data types, application scenarios, computing platforms, etc. It is necessary to classify and summarize these methods to provide a reference for the research community. In this survey, we present a comprehensive overview of recent advances in deep learning-based image inpainting. First, we categorize the deep learning-based techniques from multiple perspectives: inpainting strategies, network structures, and loss functions. Second, we summarize the open source codes and representative public datasets, and introduce the evaluation metrics for quantitative comparisons. Third, we summarize the real-world applications of image inpainting in different scenarios, and give a detailed analysis on the performance of different inpainting algorithms. At last, we conclude the survey and discuss about the future directions.

1. Introduction

In the past few decades, digital images have become one of the most important carriers of information recording and dissemination. Various image processing technologies have been proposed to meet the requirement of image-based applications, such as image denoising, image super-resolution, image colorization, image inpainting, and so on. Among them, image inpainting aims to provide a visually plausible restoration for the missing regions of damaged images. Due to different reasons, images get corrupted or some of their regions go missing. If the missing regions are small, such as regular 2-by-2 sampling patterns, image interpolation can solve the problem; otherwise, image inpainting is required. For example, a scanned old photo with cracks, a captured image with unwanted objects, a mural image with damaged paintings, etc.

Image inpainting is a very challenging problem. First, the inputs are very complex. Except for the traditional gray and color images of nature scenes [1–3], line drawings/sketches [4,5], textures [6], texts [7] and depth images [8–10] are also common and important inputs. Different types of inputs may lead to different inpainting strategies or algorithms. Second, the damage to the images may be very large, which commonly leads to unsatisfactory results for traditional patch-based algorithms [11–13], partial differential equations-based algorithms [2,3,14], or interpolation algorithms [15,16]. Third, inpainting is an ill-posed problem, which means the inpainting results are not unique, while most algorithms consider only one possible result [17,18].

With the rapid development of deep learning in computer vision, deep learning-based algorithms demonstrate high effectiveness in inpainting tasks. Compared to traditional algorithms, deep learning-based algorithms can capture better high-level semantics and obtain significantly improved results [19,20]. Although these methods have many similarities, they also have different characteristics due to the varieties of data types, application scenarios, and computing platforms, etc. It is necessary to classify and summarize these methods to provide a reference for the research community. We track the development of deep learning-based inpainting

* Corresponding author.
E-mail address: qzou@whu.edu.cn (Q. Zou).

https://doi.org/10.1016/j.patcog.2022.109046
0031-3203/

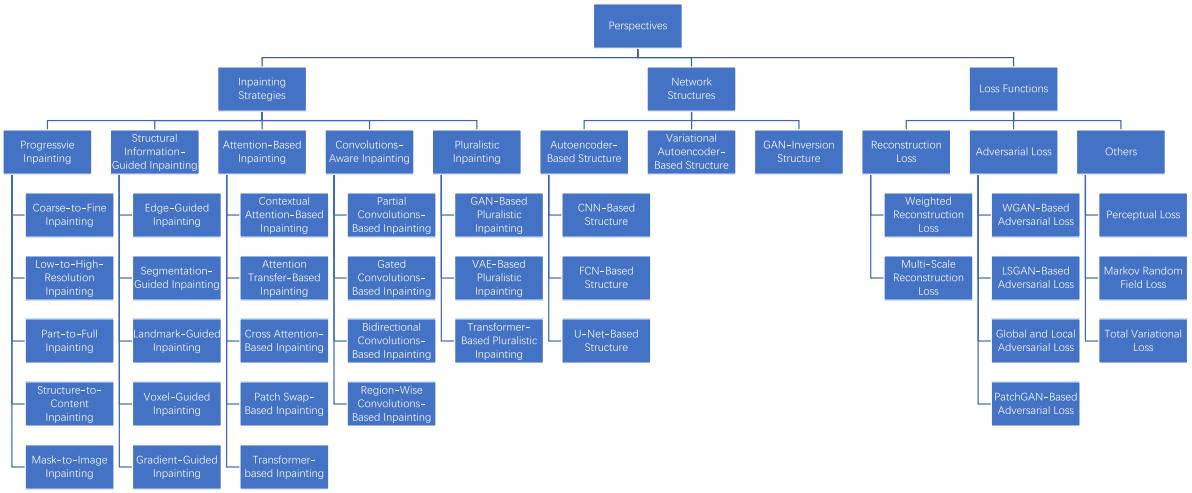


Fig. 1. Categorization of deep learning-based methods from multiple perspectives.

algorithms and select the state-of-the-arts and typical ones that present new solutions to the inpainting problems as the review focus. The new solutions could be the use of different inpainting strategies, or the modifications in network structures and loss functions. First, we consider inpainting strategies. For example, inpainting in a progressive fashion is a common-used mode [21,22]. Structural information, attention modules, and special convolutions are also worthy of attention when the missing regions become large and complex in damaged images [21–23]. Moreover, pluralistic inpainting is a new research direction for the ill-posed problem [17]. Second, we explore the development of network structures. As the research further develops, complex networks are progressively applied, from autoencoders [1,23,24] to variational autoencoders [17] and GAN-inversion networks [25]. Third, we focus on the use of loss functions in the training process. Among them, reconstruction loss and adversarial loss are two basic loss functions [1], and get improved by the later algorithms [21,24]. Based on the original design of the state-of-the-arts and typical algorithms, we survey extended algorithms to indicate how these design works better. In this paper, we choose 42 deep learning-based methods in total for evaluation, and categorize them from three perspectives: inpainting strategies, network structures, and loss functions. Fig. 1 illustrates the framework.

1.1. Inpainting strategies

Inpainting strategies present different solutions to the problems of inpainting. According to different problems, we classify the deep learning-based algorithms into progressive, structural information-guided, attention-based, convolutions-based, and pluralistic inpainting.

Progressive inpainting fills images in a step-wise manner, because inpainting tasks are essentially a puzzle solving process that is hard to accomplish in an action. Especially when the missing regions become larger, visible information is not sufficient for recovering all the pixels in one step. Progressive inpainting covers five types: coarse-to-fine, part-to-full, low-to-high-resolution, structure-to-content and mask-to-image inpainting. Yu et al. [21], Sagong et al. [26] and Guo et al. [27] employ the coarse-to-fine strategy, which makes a coarse prediction first, and then takes the coarse prediction as input to predict more refined results. Li et al. [28], Zhang et al. [29], and Zeng et al. [30] exploit the part-to-full strategy, inferring hole boundaries and then using the results as clues for further inference. Yang et al. [31] and Yi et al. [32] fill the missing regions of damaged images at each scale and upsample the images for the next scale, so we call them low-to-high-resolution inpainting. Nazeri et al. [22], Liao et al. [33], and Xiong et al. [34] adopt the structure-to-content strategy, which hallucinates the structure of missing regions, and fills them using the hallucinated structure as a priori. The last one is the mask-to-image strategy. Wang et al. [35] first estimate where to fill and then generate what to fill.

Structural information-guided inpainting relies on the structure of the known regions, such as edges, segmentation, etc. When damaged images contain sharp details, the lack of fine structure in the missing regions is a giveaway that something is amiss [22]. Typical structural information-guided inpainting covers five types: edge, segmentation, landmark, voxel and gradient guided inpainting. It is worth noting that occasionally structural information-guided inpainting can also be progressive, however, others demonstrate a small difference: structural information serves as guidance in training. We focus on the latter situation.

Attention-based inpainting considers the information from distant spatial locations, to solve the problem encountered by inpainting tasks where CNNs are not effective for borrowing distant features. According to different attention mechanisms, attention-based inpainting presents five different types: contextual attention-based, attention transfer-based, cross attention-based, patch swap-based, and transformer-based inpainting. Yu et al. [21], Sagong et al. [26] and Wang et al. learn to borrow feature information from contextual patches to generate missing regions, so we call this contextual attention-based inpainting. Zeng et al. [36], Yi et al. [32], and Zeng et al. [30] employ the attention transfer-based strategy, which obtain attention scores through another feature map to guide the missing patches generating. Zhao et al. [18] propose the cross attention-based strategy, computing the attention scores of instance patches with contextual patches to guide the generation of missing regions. Song et al. [37], Liu et al. [38], Wang et al. [39], and Wang et al. [40] exploit the patch swap-based strategy, where each patch literally searches for the most similar patch, and swaps with that patch. Wan et al. [41] and Yu et al. [42] adopt transformers [43], a more complex attention mechanism, to model the underlying distribution of reconstructed images.

Convolutions-aware inpainting employs masks to indicate the missing regions [44] and control the way how information is propagated across multiple regions [45,46]. As for damaged images, not all the information is useful. Traditional convolutions are conditioned on both valid pixels as well as substitute values in the missing regions, leading to artifacts such as color discrepancy and blurriness [23]. Liu et al. [23] propose the partial convolutions-based

strategy, automatically updating masks to distinguish the missing regions. However, partial convolutions-based inpainting updates the mask with hard rules, which would limit the flexibility. Yu et al. [47] provide a learnable mechanism for the mask updating, named gated convolutions-based inpainting. Besides, partial convolutions-based inpainting only updates masks in the encoding network, ignoring the decoding network. Xie et al. [48] employ the bidirectional convolutions-based strategy, not only using a learnable attention map module for the mask updating, but also implementing it in the decoding network. The last is the region-wise convolutions-based strategy. Ma et al. [49] do not design new convolutions, but treat the known and missing regions with different convolution filters. For convolutions-aware inpainting, a critical issue is how to generate irregular holes. Accordingly, holes generation algorithms are proposed.

Pluralistic inpainting generates multiple results for a single damaged image, because inpainting is an ill-posed problem, where a number of visually plausible results can satisfy the constraints of image restoration [18]. Typical pluralistic inpainting covers three types: GAN, VAE, and transformer based pluralistic inpainting. Cai and Wei [50] and Liu et al. [51] employ GANs to generate real inpainting results and input random noise to improve the variety of the results. Zheng et al. [17], Zhao et al. [18] and Peng et al. [52] exploit variational autoencoders and GANs to generate more than one possible results. Wan et al. [41] and Yi et al. [42] adopt transformers [43] to model the underlying distribution of reconstructed images, and each sampled vector corresponds to one result.

1.2. Network structures

Concerning various network structures, deep learning-based algorithms can be broadly classified as autoencoder-based, variational autoencoder-based, and GAN-inversion structure.

Autoencoder-based structure trains a convolutional neural network to regress the missing pixel values. Typical autoencoder-based structure performs two steps: (1) an encoder capturing the context of an image into a compact latent feature representation; and (2) a decoder that uses the representation to produce the missing image content [1]. The progress of autoencoder-based structure relies on the development in image processing field. Pathak et al. [1] derive the network structure from the AlexNet [53], a classical CNN structure, which is suitable for image classification tasks. Iizuka et al. [24] refer to image segmentation tasks, transforming the fully-connected layers in the CNN structure into convolutional layers to accept input images of any size. Liu et al. [23] design U-Net-based structure from the U-Net [54], which is widely used in image segmentation [54] and image translation tasks [55].

Although existing deep learning-based algorithms are able to produce visually realistic and semantically correct results, they produce only one result for each corrupted input [18]. Zheng et al. [17] and Zhao et al. [18] adopt variational autoencoder-based structure, which sets limitations on the encoding stage to force latent vectors to roughly follow a standard normal distribution. The sampled latent vector contains information of the missing regions, and each latent vector corresponds to one result.

GAN-inversion structure is independent of autoencoder-based and variational autoencoder-based structure. Generally, GANs serve as adversarial loss in the training process. However, Yeh et al. [56], Vitoria et al. [57], Lahiri et al. [58] and Pan et al. [25] propose GAN-inversion structure that aims to find a vector in the latent space that best reconstructs the given image, where the GAN generator is learned in advance and fixed.

1.3. Loss functions

Loss functions play an important role in the process of network training. Here we briefly discuss two basic loss functions. Other special loss functions will be explored in Section 4.

Reconstruction loss is responsible for capturing the overall structure of the missing regions and contextual coherence [1]. Pixel-wise distance between the original input and the final output is computed. Also, two variants, weighted reconstruction loss and multi-scale reconstruction loss are widely used. However, reconstruction loss provides a blurry solution, failing to restore any high-frequency details. To ameliorate the problem, adversarial loss is added.

Adversarial loss tries to make predictions look real, based on GANs [59]. Briefly speaking, the learning procedure is a two-player game where the discriminator D takes both the predictions of the generator G and the ground truth samples as input, and tries to distinguish them; while G tries to fool D by producing samples that appear as real as possible [1]. With the development of GANs, the modified versions of adversarial loss, such as WGAN-based, LSGAN-based, global and local, and PatchGAN-based adversarial loss, are proposed, making the training process faster and more stable.

There are several survey papers related to image inpainting, e.g., image synthesis [60], face image inpainting [61,62], traditional inpainting [63,64], deep learning-based inpainting [65–70], and the compound of traditional inpainting and deep learning-based inpainting [71–73]. For example, Coloma et al. [70] focus on pluralistic inpainting that generate multiple results for a single damaged image, and analyse the underlying theory and the recent proposals. However, other surveys do not provide a comprehensive or structured overview of deep learning-based inpainting algorithms. In this survey, representative and advanced inpainting algorithms will be discussed from multiple perspectives, covering all components mentioned in Fig. 1.

The remainder of this survey is organized as follows. Sections 2–4 give the summarization from the perspectives of inpainting strategies, network structures, and loss functions, respectively; Section 5 introduces the codes and datasets in this field; Section 6 gives the evaluation metrics; Section 7 introduces the application scenarios; Section 8 compares the performance of different methods; Section 9 discusses on the challenges and future directions, and Section 10 concludes the survey.

2. Inpainting strategies

Early deep learning-based algorithms [1] perform well only for small and regular holes, while inpainting with specific strategies demonstrates the superiority as the case gets more complicated. In this section, we review the current inpainting algorithms from the inpainting strategy perspective: (1) progressive inpainting; (2) structural information-guided inpainting; (3) attention-based inpainting; (4) convolutions-aware inpainting; (5) pluralistic inpainting.

2.1. Progressive inpainting

This family of algorithms inpaint in a progressive fashion, dividing the inpainting tasks into several subtasks. According to the content of subtasks, progressive inpainting is classified into coarse-to-fine, part-to-full, low-to-high-resolution, structure-to-content, and mask-to-image inpainting.

2.1.1. Coarse-to-fine inpainting

Yu et al. [21] propose two-stage network architecture, as illustrated in Fig. 2. The coarse network is a simple encoder–decoder

Fig. 2. Pipeline of the coarse-to-fine network architecture [21].

backbone, which obtains a coarse result. The refine network consists of the same backbone and a similar backbone that contains an attention module, and gets a refined result. A new attempt is to unify the two-stage network into a single-stage encoder-decoder network [26]. The encoding stage is shared, while the decoding stage is a parallel network that contains both coarse and refinement paths. The two-stage network can also be inserted into a plug-and-play framework, which combines predictive filtering and GANs, where predictive filtering preserves local structure and removes artifacts, and GANs completes the missing regions [27].

2.1.2. Part-to-full inpainting

Part-to-full inpainting divides the inpainting tasks into several subtasks, and each subtask inpaints the missing regions from the outermost to the center. The intermediate results can be shared by the Long Short-Term Memory (LSTM) [29]. Zeng et al. [30] generate a confidence map to evaluate the credibility of the intermediate results, and trust only high-confidence pixels. The part-to-full process can also be executed in the feature map space rather than the image space [28].

2.1.3. Low-to-high-resolution inpainting

Low-to-high-resolution inpainting downsamples the high-resolution image to a low-resolution version and fills it. The low-resolution result can be upsampled and get inpainted again [31], or be used to guide the inpainting of the high-resolution image [32].

2.1.4. Structure-to-content inpainting

A common-used type of structure are edges. Usually, edge-to-content inpainting adopt two-stage network architecture [22,33,34]. The edge-completion network takes the uncompleted edge map as input, while the content-completion network takes both the completed edge map and the damaged image as input. However, the input of the edge-completion network is not necessarily all the edges. Xiong et al. [34] focus on foreground objects. The DeepCut [74] and the connected component analysis [75] are used to generate the uncompleted foreground edge map.

Another type of structure is segmentation, and we mainly discuss how to extract structural information. For example, Song et al. [76] use the Deeplabv3+ [77] to generate the segmentation labels for the damaged image, and Ren et al. [78] adopt the edge-preserved smooth method [79] to extract structural information. The edge-preserved smooth result remove high-frequency details while retaining sharp edges and low-frequency structure, as shown in Fig. 3.

For human faces, facial landmarks are used to describe key points. Sun et al. [80] predict landmark coordinates conditioned on image context (e.g. body pose) to guide the blackhead image or blurhead image inpainting.

2.1.5. Mask-to-image inpainting

Previous inpainting algorithms assume that the missing regions are known and indicated by masks. However, drawing masks manually is a time-consuming task and limits practical applications. Generating masks automatically is necessary. Wang et al. [35] adopt two-stage network architecture. The mask-prediction network output the the potential visually inconsistent regions of damaged images, while the image-inpainting network takes both the predicted mask and the damaged image as input.

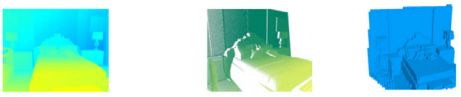
(a) Original image (b) Edge-preserved smooth image

Fig. 3. Example of the edge-preserved smooth image [78].

Discussion. Progressive inpainting strategies divide the inpainting tasks into several subtasks. Additional information obtained from the previous subtasks assists in the final result generation. For coarse-to-fine inpainting, the intermediate coarse result restores the overall structure of the missing regions, and guides the refinement network to restore local details by the attention module [21,26]. However, the two-stage network architecture significantly increases the number of model parameters and computation time. PEPSI [26] maintains a slightly better result than CA [21], and processes a single image in less time. JPG-Net [27], a plug-and-play framework, takes slightly more time than the original deep generative networks. For part-to-full inpainting, the number of intermediate results could be more. On the one hand, more intermediate results help to infer the pixels in the center of the missing regions; on the other hand, intermediate results are associated with the error accumulation. Part-to-full inpainting methods will consciously deal with the accumulation of errors. PGN [29] employs the LSTM to link the intermediate information in each subtask and RFR-Net [28] merges the intermediate results adaptively, both not relying on just a single intermediate result. ProFill [30] evaluates the credibility of the intermediate results to avoid wrong inference. However, the generation of intermediate results also leads to more computation time. For low-to-high-resolution inpainting, the inpainting process occurs on low-resolution images to reduce the cost of computation. NPS [31] downsamples images of size 512×512 three times, fills it from low-resolution to high-resolution. With the development of modern devices, from which the resolution increases up to 8000, only HiFill [32] works and spends less computation time. For structure-to-content inpainting, the selection of structural information is worth discussing. E-CE [33], EdgeConnect [22], and Foreground [34] employ edge information, while the edge information fails to guide the color generation. SPG-Net [76] exploits segmentation information, while the segmentation information confuses the final recovery if the appearance of the same semantic labels is too different. StructureFlow [78] adopts edge-preserved smooth results, which preserve both edge information and color information. The completion of the edge-preserved smooth results is easier than the original images with high-frequency details, and completed results is more

(a) Original image (b) Face parsing

Fig. 4. Example of the face parsing [82].



(a) Original image (b) Point cloud (c) Voxel

Fig. 5. Examples of the point cloud and voxel [9].

like the original images. Compared to natural scenery, human faces show a constant distribution of facial components. HeadInpainting [80] fill facial landmarks to guide the head image inpainting, where the scope of use is limited in face images only. For mask-to-content inpainting, without drawing masks manually, it is time-saving. Unfortunately, almost all methods are based on known masks. More mask-to-content inpainting algorithms are worth exploring.

2.2. Structural information-guided inpainting

The lack of fine structure in the filled regions is a giveaway that something is amiss, especially when the rest of images contain sharp details [22]. It is worth noting that structural information-guided inpainting can also be progressive, such as structure-to-content inpainting. In this section, we mainly focus on the situation where structural information serves as guidance in training. According to the type of structural information, it can be divided into edge-guided, segmentation-guided, landmark-guided, gradient-guided and voxel-guided inpainting.

2.2.1. Edge-guided inpainting

Yu et al. [47] propose a user-guided system. It extracts an edge map as reference, and based on the reference, users can draw a simple and intuitive sketch map to guide the process of inpainting. The inputs of the user-guided system are a damaged image, a mask, and a sketch map.

2.2.2. Segmentation-guided inpainting

Here we mainly consider two types of semantic segmentation, natural scenery and human faces. For natural scenery, Liao et al. [81] follow a encoder-decoder network, but the decoding stage is an interplay framework of semantic segmentation and image inpainting. Specifically, segmentation maps are generated at each scale of the decoder to guide the contextual information propagation of inpainting. For human faces, face parsing is a typical representation, as shown in Fig. 4. Li et al. [82] employ semantic parsing loss in the network that compares the parsing labels between inpainting results and the ground truth to regularize the face completion.

2.2.3. Landmark-guided inpainting

Facial landmarks describe key points on faces. Liao et al. [83] propose a collaborative encoder-decoder network, where the decoding stage outputs landmark detection, segmentation, and inpainting results simultaneously. The collaborative process ensures the guidance of structural information. Zhang et al. [84] also employ a collaborative method. The latent representations of masks and landmarks are concatenated to the latent representation of damaged images, to provide more auxiliary information.

2.2.4. Voxel-guided inpainting

In 3D computer graphics, a voxel represents a value on the regular grid in 3D space, as shown in Fig. 5. Han et al. [9] aim to inpaint depth maps. The depth maps can be converted to point cloud and volumetric occupancy grid. Then, the volumetric occupancy grid is completed by a 3D volume completion network [85], and the point cloud is reprojected to depth maps under different viewpoints and get inpainted under the guidance of the completed volumetric occupancy grid.

2.2.5. Gradient-guided inpainting

Yang et al. [86] also employ a collaborative encoder-decoder network, where the encoding stage takes both images and the corresponding gradient maps, and the decoding stage completes them simultaneously. At each scale of decoder, the completed structure is embedded into the decoding network to guide the content completion.

2.2.6. Others

Previous structural information-guided inpainting extract structure information explicitly, such as edges, segmentation, landmarks, etc. Liu et al. [87] assume that the features from deep layers contain structure semantics while the features from shallow layers contain texture details. Therefore, they extract structure and texture in different layers of the encoder. Then, both of the structure and texture features are completed, and added to the decoder as guidance.

Discussion. Structural information-guided inpainting relies on structural information to restore the missing regions and improves the inpainting performance significantly. For edge-guided inpainting, the edge is the most intuitive structural representation and it is simple for the user interaction [47]. However, the disadvantage of edges is obvious that edge information ensures the outline reliability, rather than the color consistency. Moreover, it is hard to extract or add edge information for images with high-frequency details. For segmentation inpainting, segmentation information still performs well in complex scenes. SGE-Net [81] extract segmentation information by using learning-based methods. On the one hand, deep learning-based image segmentation algorithms are developed rapidly in recent years; on the other hand, it is worth noting that the scope of use shows a little difference. These methods are trained on uncorrupted images, while they are used to extract segmentation information from damaged images in inpainting tasks. For landmark-guided inpainting, facial landmarks do describe the structure of human faces very well. CollaGAN [83] and DE-GAN [84] generate inpainting results under the guidance of landmarks. However, for human faces, not all the images contain sufficient landmark information (e.g. face-profile images). Therefore, softening the impact of pose variations in facial images needs deeper investigation. For voxel-guided inpainting, it is designed specifically for depth maps. Unfortunately, not much research is done on depth map inpainting. Thus more related research is worth exploring. For gradient-guided inpainting, the gradient not only explicitly represents texture information or high-frequency details but also inherently contains edge information. ISK [86] refers to gradient information rather than edge information, mainly because the edge information is sparse and poor in texture and details, while the gradient information is more like the original image. For the multi-task learning framework of ISK [86], the similarity of subtasks is necessary. For other structural information-guided inpainting, such as MEDFE [87], the selection of structural information is more flexible, not limited to the disadvantage of single structure.

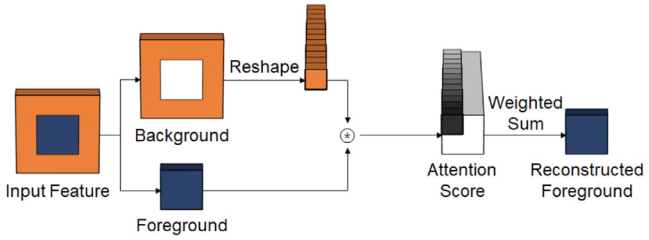


Fig. 6. Illustration of the contextual attention [21].

2.3. Attention-based inpainting

In CNNs, convolutional operations are building blocks that process one local neighbourhood at a time [88], ignoring the contributions from distant spatial locations. Wang et al. [88] refer to image denoising tasks [89] and design non-local operations to compute the response at each position by the weighted sum of the features at all positions. Following the definition of the non-local operation, it is formulated as:

$$\mathrm{y}_i = \frac{1}{C(\mathrm{x})} \sum_{\forall j} f\left(\mathrm{x}_i, \mathrm{x}_j\right) g\left(\mathrm{x}_j\right) \tag{1}$$

where i is the index of the position whose response is to be computed, j is the index that enumerates all positions, and x, y are the input and output, respectively. Here the function f computes the affinity between i and all j, and the function g computes the representation of the input. The responses are normalized by the factor $C(\mathrm{x})$. In inpainting tasks, attention-based inpainting explicitly borrows or copies information from distant spatial locations. We divide attention-based inpainting into contextual attention-based, attention transfer-based, cross attention-based inpainting, patch swap-based, and transformer-based inpainting.

2.3.1. Contextual attention-based inpainting

Yu et al. [21] add an attention module in the refinement network. The attention module computes the similarity of patches by the cosine similarity, and calculate the weighted sum of contextual information, as illustrated in Fig. 6. For each patch, especially those in the foreground, computing the weighted sum of contextual information is to copy the details from the background to the foreground. For the calculation of similarity, it can also be computed by the Euclidean distance [26]. Instead of computing the weighted sum of contextual information explicitly, Wang et al. [40] compute the similarity of patches in different decoding layers, and concatenate the attention map to the corresponding scale of the decoder to guide the inpainting process.

2.3.2. Attention transfer-based inpainting

Attention transfer-based inpainting computes the similarity of patches from another feature map, and uses the attention score to guide the process of inpainting. Zeng et al. [36] propose a pyramid-context encoder, where low-level feature maps are filled under the guidance of high-level feature maps. Then, each filled feature map is added to the corresponding scale of the decoder to guide the decoding process. For high-resolution image inpainting, Yi et al. [32] and Zeng et al. [30] apply the attention transfer module, where the attention score is learned from low-resolution images, and used to guide the filling process of high-resolution images.

2.3.3. Cross attention-based inpainting

Zhao et al. [18] introduce another technique to guide the inpainting process. The attention score is computed between instance patches and contextual patches. Then, the feature map of the instance image is reconstructed. The new feature map not only contains the features from the instance images, but also follows the contextual constraints from the damaged images.

2.3.4. Patch swap-based inpainting

Patch swap-based inpainting swaps each patch in the foreground with its most similar one in the background [37]. In the process of patch search, not only the most similar patch, but also the neighbour patch, should be considered. The patch to be swapped can be defined as the weighed sum of the two patches [38]. The patch size is also hard to decide. Wang et al. [39] use two different patch sizes to generate two feature maps. Then, the generated feature maps are concatenated by the channel equalization [90]. The new feature map is used to compute the similarity of patches.

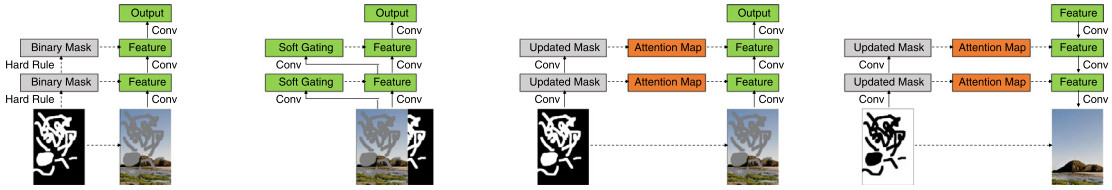
2.3.5. Transformer-based inpainting

Transformer techniques [43] have flourished in recent years. It is essentially an attention mechanism. Transformers decide for each position which other parts of the image are important [70]. The aim is to model the underlying distribution of reconstructed images. Wan et al. [41] add a transformer module in the coarse network to optimize the underlying distribution of appearance priors that contain global structure and coarse texture details. Then, the refinement network replenishes vivid texture details under the guidance of the appearance priors. Yu et al. [42] adopt a transformer module to encode global structure and high-level semantics, and an encoder to extract style features. Next, two intermediate results are used to synthesize high-resolution texture and produce inpainting results.

2.3.6. Others

The appearance flow [91] is another mechanism to consider the contributions from distant spatial locations. Ren et al. [78] calculate the correlations between the patches in the background and foreground to decide how pixels flow from the known regions to the missing regions.

Discussion. The attention mechanism is effective for borrowing or copying the information from distant spatial locations. For contextual attention-based inpainting, CA [21] computes the cosine similarity of each pair of patches. On the one hand, it is easy to compute the cosine similarity by convolutional operations; on the other hand, the cosine similarity considers only the angle between the vectors of feature patches. Obviously, the similarity cannot be fully described by the angle. Therefore, PEPSI [26] computes the similarity by the Euclidean distance, which considers not only the angle, but also the magnitude of feature patches. Contextual attention-based inpainting computes the similarity of patches in the background or foreground, but how to define the patch across both background and foreground is a problem. MA [40] employs the pixel-wise attention mechanism, where the pixels are clearly divided into the background and foreground. For attention transfer-based inpainting, the use of the attention mechanism is more flexible. For example, the attention transfer mechanism can be cross-scale. PEN-Net [36] computes the attention map from high-level feature maps and guides the low-level feature map filling. Hi-Fill [32] and ProHill [30] compute the attention map from low-resolution images, and guide the high-resolution filling. For cross attention-based inpainting, it is used in the situation where instance images are introduced. To make better use of the instance image, the feature map of the instance image needs to be reconstructed by the cross attention mechanism to follow the contextual constraints of the damaged image. For patch swap-based inpainting, the use of it is also flexible. For example, the selection of swapping patches and the generation of feature maps can be multi-scale. CSA [38] computes the patch to be swapped based on both the most similar patch and the neighbour patch to keep the



(a) Partial convolutions (b) Gated convolutions (c) Bidirectional convolutions (forward) (d) Bidirectional convolutions (reverse)

Fig. 7. Illustration of the special convolutions [47].

continuity of the generated patches. MUSICAL [39] generates feature maps based on different patch sizes to ensure both the global consistency and the local details. For transformer-based inpainting, the advantage of transformers [43] is that it can generate pluralistic results by modelling the underlying distribution of reconstructed images. We will discuss it more in the pluralistic strategy. However, transformers face the problem that the computational cost of synthesizing high-resolution images increases drastically. Therefore, ICT [41] and BAT [42] downsample the high-resolution image to a low-resolution version, and optimize the underlying distribution of it.

2.4. Convolutions-Aware inpainting

Early deep learning-based algorithms [1,24] are trained and tested on images with regular holes, while the holes can be of any shape in real applications. Convolutions-aware inpainting modifies convolution filters to adapt to the change, and can be trained and tested on images with irregular holes. According to the type of the convolution filters, they can be divided into partial convolutions-based, gated convolutions-based, bidirectional convolutions-based, and region-wise convolutions-based inpainting. To make the subsection easier to understand, we add a figure showing the visual difference between each convolution, as illustrated in Fig. 7. At the end of the subsection, we will discuss irregular holes generation methods.

2.4.1. Partial convolutions-based inpainting

Liu et al. [23] replace traditional convolutions with partial convolutional layers. The partial convolutional layer consists of a partial convolution operation and a mask update function. For the partial convolution operation, it is only conditioned on the valid pixels defined by the current mask. For the mask update function, it is updated with the hard rule that if the current window includes valid pixels, the value of the current location turns to be 1.

2.4.2. Gated convolutions-based inpainting

Yu et al. [47] propose gated convolutions that learn masks automatically from data rather than update masks using the hard rule. Based on the current mask, the gated convolution operations can be conditioned on valid pixels. Then, Yi et al. [32] propose light weight gated convolutions to reduce the number of parameters to automatically learn masks.

2.4.3. Bidirectional convolutions-based inpainting

Xie et al. [48] adopt bidirectional convolutions, containing a forward attention map module and a reverse attention map module. The forward attention map module takes the mask of the missing regions as input, while the reverse attention map module takes the mask of the known regions as input. The former masks are updated in the encoding stage, while the latter masks are reversely updated in the decoding stage. At each scale of the encoder and decoder, the corresponding mask defines the valid pixels to be conditioned on.

2.4.4. Region-wise convolutions-based inpainting

Ma et al. [49] incorporate region-wise convolutions in the decoding network. The decoder contains two streams of convolutions, one for generating content for the known regions, the other for generating content for the missing regions.

2.4.5. Irregular holes generation

Two types of methods are usually used to generate irregular holes.

The first type generates holes by randomly removing the templates of real objects. For example, Pathak et al. [1] refer to the PASCAL VOC 2012 dataset [92], in which the segmentation labels describe the shapes of real objects. Liu et al. [23] publish the NVIDIA Irregular Mask dataset [23] that collects a fixed set of irregular masks from videos.

The second type generate irregular holes by simple algorithms. Yu et al. [47] draw lines and rotate angles repeatedly, and draw circles in joints to ensure the smoothness. Xiao et al. [93] randomly blend multiple shapes including rectangles, circles, ellipses, and strings, or select a point and expand it to the surrounding regions by the morphology process of dilation to produce the final mask. Examples of the irregular masks obtained by different methods are shown in Fig. 8.

Discussion. Convolutions-aware inpainting performs well for images with irregular holes. For partial convolutions-based inpainting, the partial convolutional layers can easily be implemented in different deep learning frameworks [23]. However, the hard rule for the mask updating is simple but problematic. First, it classifies all the pixels to be either valid or invalid, not considering the number of pixels covered by convolution filters. In other words, if the current window includes valid pixels, regardless of the number, the value of the current location turns to be 1. Second, all the invalid pixels will progressively disappear in deep layers, and at this time, the partial convolutional layers are the same as traditional convolutions. For gated convolutions-based inpainting, the hard rule for the mask updating is replaced by a learning rule. However, it almost doubles the number of parameters and processing time in comparison to traditional convolutions [32]. HiFill [32] changes the calculation of gate branches and reduces the number of parameters drastically. For bidirectional convolutions-based inpainting, LBAM [48] introduces the reverse attention map module to ensure that convolution operations are always conditioned on valid pixels, even in the decoding stage. For region-wise convolutions-based inpainting, RN [49] treats the known and missing regions with different convolution filters. Intuitively, the convolution filters for the known regions are responsible for the content reconstruction, while the convolution filters for the missing regions focus on the contextual information inference.

(a) NVIDIA Irregular Mask dataset (b) Random elements blending (c) Area growing

Fig. 8. Examples of the irregular masks got by different methods [93].

2.5. Pluralistic inpainting

Inpainting is an ill-posed problem, thus this family of algorithms aim to generate more than one visually plausible results. The key to the problem is to find a vector or latent representation to restrict the generation of results. In other words, if each vector corresponds to one result, introducing different vectors can generate various results. Depending on how the vector is generated, pluralistic inpainting can be classified into GAN-based pluralistic inpainting, VAE-based pluralistic inpainting, and transformer-based pluralistic inpainting.

2.5.1. GAN-based pluralistic inpainting

The diversity of GAN-based pluralistic inpainting is kept by random vectors. Cai and Wei [50] follow a two-path network in the training stage. The first path introduces an extractor to estimate the style feature of the ground truth. Then, the extracted feature is added to the damaged image as input. The second path adds a random vector to the damaged image, and inpaints it by a deep generative network. Liu et al. [51] propose a coarse-to-fine network. The coarse network obtains a coarse result, while the refinement network takes a random vector as input and modulate it under the guidance of the coarse result at each scale of the decoder.

2.5.2. VAE-based pluralistic inpainting

Variational autoencoders encode damaged images into a latent distribution, from which the sampled vectors keep the diversity. Zheng et al. [17] propose a two-path network. The first path takes damaged images as input, and learns the latent representation and distribution to generate inpainting results. The second path takes the complement of damaged images as input, and also learns the latent representation and distribution. Then, the sampled vector that contains the information of the missing regions is concatenated to the sampled vector in the first path that contains the information of the known regions, and the new vector contains enough information to reconstruct the original image. Zhao et al. [18] follow an encoder-decoder network. The encoding stage is a parallel network that takes damaged images and instance images as input, and learns the latent representations, respectively. The cross attention module combines the information of the representations, and sends it to a shared decoder. Peng et al. [52] train a variational autoencoder in advance to disentangle images into discrete structural features and textural features. Then, autoregressive models are used to learn the latent distribution of the damaged image, which is constrained by the above structural features. Finally, a deep generative network takes the damaged image as input and generates pluralistic results under the guidance of the sampled vectors from the latent distribution.

2.5.3. Transformer-based pluralistic inpainting

We have discussed transformers [43] in the attention-based strategy. Transformers are used to model the underlying distribution of reconstructed images. ICT [41] and BAT [42] are two typical transformer-based pluralistic inpainting. To reduce the computational cost caused by transformers, ICT [41] and BAT [42] downsample the image resolution from high to low, and optimize the underlying distribution of it.

Discussion. Pluralistic inpainting generates more than one possible results for damaged images. For GAN-based pluralistic inpainting, the diversity is controlled by random vectors. PII-GAN [50] adds a random vector to the damaged image, while PD-GAN [51] generates inpainting results from a random vector. However, the random vector has nothing to do with the original image, thus the GAN-based approaches are unstable, especially for large and complex scenes. For VAE-based pluralistic inpainting, damaged images are encoded into a latent representation and distribution. Based on the distribution, each sampled vector corresponds to one inpainting result. However, the training of variational autoencoders is also not stable [17]. Therefore, PIC-Net [17] and UCT-GAN [18] learn the other latent distribution from the ground truth or instance images to restrict the distribution of damaged images, and DSI-VQVAE [52] follows the VQ-VAE [94] to generate discrete features, for the stable training. For transformer-based pluralistic inpainting, transformers [43] are adopted to model the underlying distribution of reconstructed images. ICT [41] and BAT [42] employ transformers to restore the global structure of damaged images, and produce texture details under the guidance of it. Considering the quadratically increasing computational cost of the multi-head attention in transformers, ICT [41] and BAT [42] extract the structure feature from the low-resolution version of damaged images. It is reported in [70] that the diversity of the inpainting results generated by GAN-based and VAE-based pluralistic inpainting is still limited, while transformer-based methods have achieved better performance.

3. Network structures

Network structure is the core of inpainting algorithms. Essentially, it follows an encoder-decoder pipeline: (1) an encoder capturing the context of an image into a compact feature representation; and (2) a decoder using the representation to generate the missing content [1]. In this section, we classify inpainting algorithms from the network structure perspective as follows: (1) autoencoder-based structure; (2) variational autoencoder-based structure; and (3) GAN-inversion structure.

3.1. Autoencoder-based structure

Autoencoders learn the latent representation of a set of data in an unsupervised manner [95]. Given an input x, the purpose is to reconstruct the output. Following the definition of the autoencoder, it is formulated as:

$$\phi, \psi = \arg\min_{\phi,\psi} \|\mathrm{x} - (\psi \circ \phi)(\mathrm{x})\| \tag{2}$$

where ϕ, ψ denote the encoder and decoder, respectively. In inpainting tasks, autoencoders use the representation to not only reconstruct itself, but also generate the missing content. Next, we introduce some representative network structures in this family.

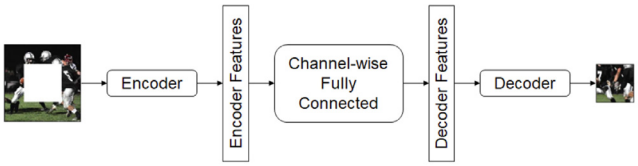


Fig. 9. Network of the CNN-based structure [1].

3.1.1. CNN-Based Structure

CNNs are widely used in image classification tasks. In inpainting tasks, Pathak et al. [1] refer to the AlexNet [53], a encoder-decoder network, but use fully-connected layers to link encoder features and decoder features. The network structure is illustrated in Fig. 9. To improve the inpainting results, the activation functions in [1] can be replaced with the ELU activation [31] to make the network training more stable, and the deconvolutional layers in the decoder can also be replaced with upsampling layers and convolutional layers to avoid checkerboard artifacts [96].

3.1.2. FCN-Based Structure

FCNs are typical network structure in image segmentation tasks. In inpainting tasks, Iizuka et al. [24] transform the fully-connected layers [1] into convolutional layers. Also, they insert dilated convolutional layers [97] into the network to obtain a larger receptive field for the contextual information propagation. The dilated convolutional layers can be incorporated with the residual blocks [98] to provide better learning capacity [22,76,99]. In each residual block, progressively increasing the dilated factors of the dilated convolutional layers helps further expand the receptive field [76].

3.1.3. U-Net-based structure

The main feature of U-Net-based structure is the skip links that concatenate the feature maps in the encoding network to the decoding network. Liu et al. [23] derive the network structure from the U-Net [54]. The convolutional layers can be replaced with the NIN [100] to provide multiple receptive fields [93]. For the consistency inside and outside masks, Hong et al. [101] introduce fusion blocks that generate alpha composition maps to blend the known and missing regions, and embed them to the decoding stage to guide the process of inpainting.

Discussion. Most inpainting algorithms are derived from CNNs, FCNs and U-Nets. For CNN-based structure, CE [1] employs the fully-connected layers to propagate the contextual information. It is worth noting that the fully-connected layers is in a channel-wise manner to avoid the explosion in the number of parameters. NPS [31] exploits the ELU activation functions for the stable training, and SI [96] adopts the combinations of upsampling layers and convolutional layers to avoid checkerboard artifacts. However, CNN-based structure faces the problem that the size of input images is limited, due to the fully-connected layers. For FCN-based structure, the size of input images can be arbitrary. GL [24] adopt the dilated convolutional layers to obtain a larger receptive field for the contextual information propagation. EdgeConnect [22] and SPG-Net [76] incorporate them with the residual blocks [98] to provide better learning capacity. Moreover, SPG-Net [76] progressively increases the dilated factors in each residual block to further expand the receptive field. For U-Net based structure, the skip links concatenate the encoder features of the known regions to the decoder, making it possible for the decoder to copy valid pixels and focus on the missing regions. PartialConv [23] first adopts the U-Net-based structure in inpainting tasks. Since then, U-Net-based structure is widely used in convolution-aware inpainting. The use of U-Net-based structure can also be more flexible. DI-Net [93] refers to the NIN [100] to provide multiple receptive fields, considering local details, global structure, and invariant features, simultaneously. DF-Net [101] introduces the fusion blocks to ensure the consistency inside and outside masks. However, the effect of the skip connections is still controversial. It is reported in [47] that the skip connections have no significant effect, while Xie et al. [48] claim that the skip connections perform well. Therefore, more investigation about autoencoder-based structure is needed.

3.2. Variational autoencoder-based structure

Variational autoencoders not only learn the representation of a set of data, but also make strong assumptions concerning the distribution of the latent variables. Assuming that the data is generated by the model $p_\psi(x|z)$, and the encoder learns the approximation $q_\phi(z|x)$ to the posterior distribution $p_\psi(z|x)$, where ϕ, ψ denote the encoder and decoder respectively. The training objective of variational autoencoders has the following form:

$$\mathcal{L}(\phi,\psi,x) = D_{KL}\left(q_\phi(z|x)\|p_\phi(z)\right) - \mathbb{E}_{q_\phi(z|x)}\left(\log\left(p_\psi(x|z)\right)\right) \tag{3}$$

where D_{KL} denotes the KL divergence. Note that the prior over the latent variables is usually set to be the Gaussian distribution:

$$p_\psi(z) = \mathcal{N}(0, I) \tag{4}$$

As we have discussed before, variational autoencoder-based structure generates multiple plausible solutions. PIC-Net [17], UCT-GAN [18], and DSI-VQVAE [52] are typical variational autoencoder-based structure. To stabilize the training process, PIC-Net [17] and UCT-GAN [18] learn the other latent distribution from the ground truth or instance images to restrict the distribution of damaged images, and DSI-VQVAE [52] follows the VQ-VAE [94] to generate discrete features.

Discussion. Variational autoencoder-based structure infers the latent distribution of damaged images, and samples from it to generate diverse results. However, the diversity of the inpainting results based on a single ground truth is intrinsically limited [102]. In other words, if the ground truth image is without glasses, all of the inpainting results will be without glasses. Obviously, the inpainting results with glasses are theoretically possible. Moreover, the deep understanding of the latent distribution is lacking. Taking human faces for example, the sampled latent vector corresponds to the inpainting result whose pupils' color is brown, but we cannot modify the sampled latent vector to change the color to another. Generating truly diverse results is still challenging.

3.3. GAN-Inversion Structure

GAN-inversion structure trains a fixed GAN generator in advance and aims to find a vector in the latent space which is best to reconstruct the damaged image [25]. Assuming that the input image $\mathtt{I}_{in}$ is obtained via $\mathtt{I}_{in} = \phi\left(\mathtt{I}_{gt}\right)$, where $\mathtt{I}_{gt}$ is the corresponding ground truth and ϕ is the degradation transformation. The training objective of GAN-inversion structure has the following form:

$$\hat{z} = \arg\min_{z\in\mathbb{R}^d}\left(\mathcal{L}_{rec}(\mathtt{I}_{in}, \phi(G(z;\theta)))\right) \tag{5}$$

where z denotes vectors in the latent space, and G represents the GAN generator parameterized by θ. Then, we get the network prediction $\mathtt{I}_{out}$:

$$\mathtt{I}_{out} = G\left(\hat{z};\theta\right) \tag{6}$$

Fig. 10 illustrates the basic GAN-inversion structure [56,57]. Lahiri et al. [58] train an extractor to predict the latent vector of the damaged image to reduce the computational cost. Pan et al. [25] propose relaxed GAN-inversion structure that fine-tunes the parameters of the GAN generator and latent vector simultaneously to narrow the gap between natural images and actual ones.

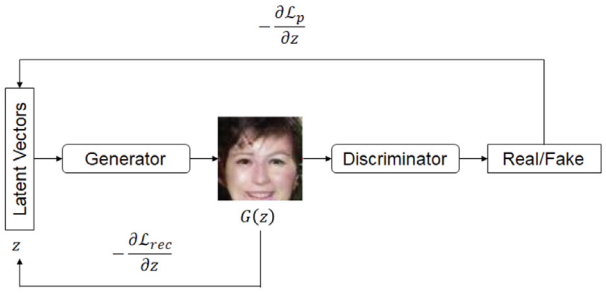


Fig. 10. Network of the GAN-inversion structure [56].

Discussion GAN-inversion structure is different from the previous network structure that follows an encoder-decoder pipeline. The advantages of GAN-inversion structure is that it can be used more flexibly. First, the degradation transformation can be in any form, such as graying transformation, corrupting transformation, and downsampling transformation [25]. In other words, the application scope of GAN-inversion structure is not limited to inpainting tasks, and it can also be used in colorization and super-resolution tasks. Second, different types of GANs can be used in GAN-inversion structure to adapt to different application scenarios [58]. That is to say, the progress of GANs also promotes the development of GAN-inversion structure. However, GAN-inversion structure faces two key problems. First, SI [56] directly searches for the closest vector of the damaged image, leading to the high computational cost. PG-GAN [58] learns to predict the latent vector to reduce the computational cost. Second, there is a gap between the approximated manifold of natural images and the actual ones [103]. DGP [25] fine-tunes the parameters of the GAN generator to adapt to the gap. Although the gap can be fine-tuned, GAN-inversion structure is still limited to bounded datasets where the manifold and diversity of images is simple, such as faces, facades, etc. For general datasets, the performance of GAN-inversion structure is really poor.

4. Loss functions

Loss functions, the training objective of networks, penalize the deviation between network predictions and true data labels. In this section, we review common-used loss functions in inpainting tasks. There are two basic loss functions: (1) reconstruction loss; and (2) adversarial loss.

4.1. Reconstruction loss

Reconstruction loss computes the pixel-wise distance between the network prediction $\mathtt{I}_{out}$ and the ground truth image $\mathtt{I}_{gt}$:

$$\mathcal{L}_{rec} = \|(1-\mathtt{M}) \odot (\mathtt{I}_{gt} - \mathtt{I}_{out})\|_2 \tag{7}$$

where $\mathtt{M}$ is the binary mask (0 for holes), and $\odot$ is the element-wise product operation.

4.1.1. Weighted reconstruction loss

Weighted reconstruction loss pays more attention to the missing regions that are close to the boundaries of holes. The weighted terms can be pixel-wise. Yeh et al. [56] assign weight to each pixel according to the number of the known pixels surrounding it. Lahiri et al. [58] compute the distance between each pixel and the boundary as the weight. Wang et al. [104] set the confidence of the known pixels as 1 and iteratively propagate the confidence to the missing regions by the Gaussian filter. The weighted terms can also be region-wise, which assign smaller constant weight to all pixels close to the center of holes [32].

4.1.2. Multi-scale reconstruction loss

Multi-scale reconstruction loss progressively refines the network prediction for the missing regions at each scale of the decoder. Liao et al. [81], Zeng et al. [36], and Hong et al. [101] transform the feature maps in the decoding stage into RGB images, and proportionally resize the ground truth images to the same size, to compute reconstruction loss at each scale of the decoder.

Discussion Reconstruction loss aims to capture the overall structure of the missing regions and contextual coherence. For general reconstruction loss, CE [1] computes the distance using ℓ_1 or ℓ_2 distance. However, general reconstruction loss treats each pixel equally, while the strong enforcement in the central pixels has a bad effect on the inpainting results. For weighted reconstruction loss, it considers that the missing pixels near the boundaries of holes have much less ambiguity than those close to the center of holes, and introduces the weighted term. SI [56], PG-GAN [58], and GMCNN [104] compute the weight for each pixel according to the distance between the pixel and the boundaries of holes. However, the pixel-wise weighted term is computationally expensive, especially for high-resolution images. HiFill [32] treats all the central pixels with smaller constant weight. For multi-scale reconstruction loss, it progressively restricts the generation of results, helping to capture the overall structure.

4.2. Adversarial loss

Adversarial loss is derived from GANs [59]. GANs consist of the generator G and the discriminator D. Both G and D are deep networks, where G maps samples from the noise distribution $\mathcal{Z}$ to the real data distribution $\mathcal{X}$, while D provides loss gradients to the generative network. The learning procedure is like a two-player game, where D takes both the network predictions of G and the ground truth samples as input and tries to distinguish them, while G tries to fool D by generating samples that appear as realistic as possible [1]. We define it as:

$$\min_G \max_D \left(\mathbb{E}_{x\sim\mathbb{P}_r}[\log(D(x))] + \mathbb{E}_{\hat{x}\sim\mathbb{P}_g}\left[\log\left(1-D\left(\hat{x}\right)\right)\right]\right) \tag{8}$$

where $\mathbb{P}_r$ is the real data distribution and $\mathbb{P}_g$ is the generated distribution defined by $\hat{x} = G(z)$, $z \sim p(z)$. The DCGAN [105] has recently shown encouraging results in the generative modelling of images. In inpainting tasks, we enforce the generator to generate the inpainting results based on damaged images. Given the input image $\mathtt{I}_{in}$ and the ground truth image $\mathtt{I}_{gt}$, we define adversarial loss as:

$$\mathcal{L}_D = -\mathbb{E}_{\mathtt{I}_{gt}\sim p(\mathtt{I}_{gt})}[\log(D(\mathtt{I}_{gt}))] - \mathbb{E}_{\mathtt{I}_{in}\sim p(\mathtt{I}_{in})}[\log(1-D(G(\mathtt{I}_{in})))] \tag{9}$$

$$\mathcal{L}_G = -\mathbb{E}_{\mathtt{I}_{in}\sim p(\mathtt{I}_{in})}[\log(D(G(\mathtt{I}_{in})))] \tag{10}$$

In this subsection, we review the modified versions of GANs that make the training process faster and more stable.

4.2.1. WGAN-based adversarial loss

The WGAN uses the Earth-Mover distance to compare the generated distribution and the real data distribution. The training objective of the WGAN is constructed by the Kantorovich-Rubinstein duality [106]. To better enforce the Lipschitz constraint in the WGAN, the WGAN-GP [107] introduces a gradient penalty term. In inpainting tasks, Yu et al. [21] apply the gradient penalty only to the pixels in the missing regions.

4.2.2. LSGAN-based adversarial loss

The LSGAN [108] replaces the cross-entropy with the least-squares to measure the training objective. Zheng et al. [17] employ LSGAN-based adversarial loss for the the generative path. Liu et al.

[87] and Liu et al. [38] combine the LSGAN [108] and the RaGAN [109], namely RaLSGAN-based adversarial loss.

4.2.3. Global and local adversarial loss

Iizuka et al. [24] employ two discriminators in the network. One takes the whole image as input, while the other takes the patch centred around the missing regions as input.

4.2.4. PatchGAN-based adversarial loss

The PatchGAN classifies whether each $N \times N$ (e.g. 7×7) patch of the whole image is real or fake and averages all the responses to provide the final output of the discriminator [55]. Yu et al. [47] and Sagong et al. [26] adopt the spectral normalization [110] in the PatchGAN.

Discussion. Adversarial loss is responsible for generating realistic images. For WGAN-based adversarial loss, it introduces the gradient penalty to enforce the Lipschitz constraint. It is worth noting that the gradient penalty is actually based on the ℓ_1 distance, which is the same as that in reconstruction loss. The combination of WGAN-based adversarial loss and reconstruction loss not only helps generate better inpainting results, but also makes the training process faster and more stable. For LSGAN-based adversarial loss, it also aims to improve the training process. For global and local adversarial loss, GL [24] trains a global discriminator and a local discriminator. The global discriminator assesses whether the whole image is coherent, while the local discriminator ensures the local consistency of the missing regions. However, the local discriminator handles holes of limited size. In other words, if the missing regions spread over the whole image, it will be hard to find the patch of appropriate size to fit the local discriminator. For PatchGAN-based adversarial loss, it divides the whole image into patches, and accesses each patch separately. Therefore, PatchGAN-based adversarial loss can handle any types of holes. GatedConv [47] and PEPSI [26] introduce the spectral normalization [110] to stabilize the training process of the PatchGAN.

4.3. Others

Reconstruction loss and adversarial loss are widely used in inpainting tasks. In this subsection, we introduce other auxiliary loss functions.

4.3.1. Perceptual loss

Perceptual loss focuses on the difference between the high-level representations of images. Johnson et al. [111] defines two types of perceptual loss functions: (1) feature reconstruction loss; and (2) style reconstruction loss. Feature reconstruction loss computes the pixel-wise distance between the network prediction and the ground truth image at the feature level, while style reconstruction loss computes the distance between the Gram matrices. Most inpainting algorithms employ the VGG-16 [112] and the VGG-19 to compute feature reconstruction loss and style reconstruction loss. Moreover, Song et al. [76] compute perceptual loss based on the AlexNet [53]. Nazeri et al. [22] and Song et al. [76] exploit the discriminator to extract the feature maps. Dolhansky and Ferrer [113], Yan et al. [114] and Zhao et al. [18] project instance images to the latent space, thus the projection function can be used to measure the perceptual difference.

4.3.2. Markov random fields (MRF) loss

MRF loss also measures the difference between the high-level representations of images, but not using the ℓ_1 or ℓ_2 distance. For each patch in the missing regions, MRF loss computes the distance between the patch and the nearest neighbour in the known regions. Yang et al. [31] adopt the Euclidean distance to find the nearest neighbour for each patch. However, the search mode tends to produce smooth structure, where patches in the missing regions look similar. Therefore, the Euclidean distance can be replaced with the relative distance [104]. If patches in the missing regions are close to only one patch in the known regions, the relative distance will be long; if patches in the missing regions are close to different patches in the known regions, the relative distance will be short.

4.3.3. Total variation (TV) loss

TV loss, also known as the total variation regularization, is widely used in image denoising. TV loss computes the difference between the adjacent pixels in the missing regions.

Discussion. Reconstruction loss and adversarial loss are widely used in inpainting tasks. However, the inpainting results with low reconstruction loss may look perceptually worse than those with slightly higher reconstruction loss. Therefore, other auxiliary loss functions are needed. For perceptual loss, two types of perceptual loss functions are discussed. Feature reconstruction loss compares images at the feature level. It is reported in [115] that minimizing feature reconstruction loss for shallow layers tends to produce visually indistinguishable images, while minimizing feature reconstruction loss for deep layers preserves overall content and structure. However, feature reconstruction loss does not consider the style of images, such as color, texture, and exact shapes. Therefore, Gatys et al. [116] propose style reconstruction loss to penalize the difference in style. For MRF loss, NPS [31] finds the nearest neighbour for each patch, but the candidates may be the same. To diversify the structure, GMCNN [104] encourages each patch in the missing regions to find different candidates. For TV loss, it aims to ensure the spatial smoothness in the inpainting results. However, the effect of auxiliary loss is still controversial. It is reported in [21] that perceptual loss and TV loss do not bring noticeable improvements for the inpainting results, while Liu et al. [23] present the ablation study of perceptual loss and TV loss to verify the necessity. Therefore, more trials and verifications on loss functions are required.

Summary. We summarize the representative algorithms, descriptions, advantages, and disadvantages from multiple perspectives in Appendix A.

5. Codes and datasets

Open source codes are made freely available for possible modification and redistribution. Based on the codes, the original algorithms can be applied in relative tasks. In Table 1, we summarize the current image inpainting algorithms whose source codes are open.

Public datasets are the integral part of machine learning. In inpainting tasks, the categories of public datasets include objects, scenes, faces, etc. In this section, we introduce some representative datasets in each category. Table 2 summarizes the public datasets used in inpainting algorithms.

ImageNet [117] is an image dataset organized according to the WordNet hierarchy. Each subset of ImageNet represents a meaningful concept. The current version of the dataset contains 21,841 non-empty subsets and 14,197,122 images.

Paris StreetView [118] is a dataset collected from the Google StreetView of Paris. The dataset mainly focuses on the buildings in the city. It contains 14,900 training images and 100 test images.

Places [119] is a dataset of the different categories of scenes. In total, Places dataset contains more than 10,000,000 images coming from 400 scene categories. Places2 Challenge is developed based on Places.

CelebA [120] is a large-scale face attribute dataset. The dataset has large diversities, large quantities, and rich annotations, including 10,177 identities, 202,599 face images, 5 landmark locations, and 40 attributes annotations per image.

Table 1
Open source codes for inpainting.

Algorithms	Official Codes	Public Datasets
CE~[1] CVPR 2016	https://github.com/pathak22/context-encoder	ImageNet/Paris StreetView
GFC~[82] CVPR 2017	https://github.com/Yijunmaverick/GenerativeFaceCompletion	CelebA
NPS~[31] CVPR 2017	https://github.com/leehomyc/Faster-High-Res-Neural-Inpainting	ImageNet/Paris StreetView
SI~[56] CVPR 2017	https://github.com/ChengBinJin/semantic-image-inpainting	CelebA/SVHN/Stanford Cars
CA~[21] CVPR 2018	https://github.com/JiahuiYu/generative_inpainting/tree/v1.0.0	ImageNet/Places2/CelebA/CelebA-HQ/DTD
ExGAN~[113] CVPR 2018	https://github.com/bdol/exemplar_gans	Celeb-ID
PEPSI~[26] CVPR 2019	https://github.com/Forty-lock/PEPSI-Fast_image_inpainting_with_parallel_decoding_network	ImageNet/Places2/CelebA
PEN-Net~[36] CVPR 2019	https://github.com/researchmm/PEN-Net-for-Inpainting	Places2/CelebA-HQ/DTD/Facade
PIC-Net~[17] CVPR 2019	https://github.com/lyndonzheng/Pluralistic-Inpainting	ImageNet/Paris StreetView/Places2/CelebA-HQ
RFR-Net~[28] CVPR 2020	https://github.com/jingyuanli001/RFR-Inpainting	Paris StreetView/Places2/CelebA
HiFill~[32] CVPR 2020	https://github.com/Atlas200dk/sample-imageinpainting-HiFill	Places2/CelebA-HQ/DIV2K
DSI-VQVAE~[52] CVPR 2021	https://github.com/USTC-JialunPeng/Diverse-Structure-Inpainting	ImageNet/ Places2/CelebA-HQ
EdgeConnect~[22] ICCV 2019	https://github.com/knazeri/edge-connect	Paris StreetView/Places2/CelebA
StructureFlow~[78] ICCV 2019	https://github.com/RenYurui/StructureFlow	Paris StreetView/Places2/CelebA
GatedConv~[47] ICCV 2019	https://github.com/JiahuiYu/generative_inpainting	Places2/CelebA-HQ
CSA~[38] ICCV 2019	https://github.com/KumapowerLIU/CSA-inpainting	Paris StreetView/Places2/CelebA
LBAM~[48] ICCV 2019	https://github.com/Vious/LBAM_Pytorch	Paris StreetView/Places2
ICT~[41] ICCV 2021	https://github.com/raywzy/ICT	ImageNet/Places2/FFHQ
PartialConv~[23] ECCV 2018	https://github.com/NVIDIA/partialconv	ImageNet/Places2/CelebA-HQ
ProFill~[30] ECCV 2020	https://zengxianyu.github.io/iic/	Places2
VC-Net~[35] ECCV 2020	https://github.com/shepnerd/blindinpainting_vcnet	ImageNet/Places2/CelebA-HQ/FFHQ
DGP~[25] ECCV 2020	https://github.com/XingangPan/deep-generative-prior	ImageNet
MEDFE~[87] ECCV 2020	https://github.com/KumapowerLIU/Rethinking-Inpainting-MEDFE	Paris StreetView/Place2/CelebA
PGN~[29] MM 2018	https://github.com/crashmoon/Progressive-Generative-Networks	ImageNet/Paris StreetView
DF-Net~[101] MM 2019	https://github.com/hughplay/DFNet	Places2/CelebA
BAT~[42] MM 2021	https://github.com/yingchen001/BAT-Fill	Paris StreetView/Places2/CelebA-HQ
ISK~[86] AAAI 2020	https://github.com/YoungGod/sturcture-inpainting	Places2/CelebA/Facade
RN~[130] AAAI 2020	https://github.com/geekyutao/RN	Places2/CelebA
GL~[24] SIGGRAPH 2017	https://github.com/satoshiiizuka/siggraph2017_inpainting	Places2/CelebA
GMCNN~[104] NIPS 2018	https://github.com/shepnerd/inpainting_gmcnn	ImageNet/Paris StreetView/Places2/CelebA/CelebA-HQ
MUSICAL~[39] IJCAL 2019	https://github.com/wangning-001/MUSICAL	Paris StreetView/Places2/CelebA
PII-GAN~[50] IEEE Access 2020	https://github.com/vivitsai/PiiGAN	CelebA
JPG-Net~[27] Arxiv 2021	https://github.com/tsingqguo/jpgnet	Places2/CelebA

CelebA-HQ [121] is developed by a GAN model, constructing a high-quality version of CelebA. The dataset contains 30,000 images of size 1024×1024.

Helen [122,123] is originally derived from annotated Flickr images. The resulting dataset consists of 2000 images for training and 330 images for testing, with highly accurate, detailed, and consistent annotations of facial components. Based on the original Helen dataset, Smith et al. [122] annotate 11 segmentation labels covering main facial components.

DTD [124] is a texture database, collecting textural images in the wild. The dataset consists of 5640 images, organized according 47 categories.

FFHQ is a high-quality image dataset of human faces, developed by a GAN model. The dataset contains 70,000 images of size 1024×1024.

Facade [125] is a dataset of facade images from different cities with diverse architectural styles. It includes 606 rectified images of facades from various sources.

Cityscapes [126] focuses on the semantic understanding of urban street scenes. In total, Cityscapes contains 5000 annotated images with fine annotations and 20,000 annotated images with coarse annotations.

SVNH [127] is obtained from the house numbers in the Google StreetView. Specifically, SVNH consists of 99,289 images of small cropped digits.

Stanford Cars [128] is a set of car images. The dataset contain 16,185 images coming from 196 categories of cars.

Celeb-ID [113] is derived from CelebA. It contains around 17,000 identities, 100,000 face images, with at least three images of each identity.

DIV2K [129] is a high-quality dataset with a large diversity of content. The dataset contains 1000 high-resolution images of size 2000×2000. Specifically, it is divided into 800 images for training, 100 images for validation and 100 images for testing.

SUNCG [85] is a large-scale synthetic 3D scene dataset. It contains 45,622 different scenes, each with a realistic room and furniture layouts.

NVIDIA Irregular Mask [23] collects a fixed set of irregular masks. It contains 55,116 masks for training, and 12,000 masks for testing.

Foreground-aware [34] is an irregular hole mask dataset. It contains 100,000 masks for training, and 10,000 masks for testing. Each mask is a binary image of size 256×256.

6. Evaluation metrics

Evaluation metrics indicate the effectiveness of the proposed algorithms. On the one hand, mean squared error (MSE), peak signal to noise ratio (PSNR), structural similarity index (SSIM), and learned perceptual image patch similarity (LPIPS) [131] are used

Table 2
Public datasets for inpainting.

Datasets	Instances	Formats	Categories	Algorithms
ImageNet	14,197,122	Images	Objects	CE~[1]/NPS~[31]/CA~[21]/PEPSI~[26]/PIC-Net~[17]/DSI-VQVAE~[52]/ICT~[94] PartialConv~[23]/VC-Net~[35]/DGP~[25] PGN~[29]/GMCNN~[104]
Paris StreetView	15,000	Images	Streets	CE~[1]/NPS~[31]/PIC-Net~[17]/RFR-Net~[28]/EdgeConnect~[22]/StructureFlow~[78]/CSA~[38] LBAM~[48]/MEDFE~[87]/PGN~[29]/BAT~[42]/GMCNN~[104]/MUSICAL~[39]
Places	10,000,000	Images	Scenes	CA~[21]/PEPSI~[26]/PEN-Net~[36]/PIC-Net~[17]/RFR-Net~[28]/HiFill~[32]/DSI-VQVAE~[52] EdgeConnect~[22]/StructureFlow~[78]/GatedConv~[47] CSA~[38]/LBAM~[48]/ICT~[94]/PartialConv~[23] ProFill~[30]/VC-Net~[35]/MEDFE~[87]/DF-Net~[101]/BAT~[42]/ISK~[86]/RN~[130] GL~[24]/GMCNN~[104]/MUSICAL~[39]/JPG-Net~[27]
CelebA	202,599	Images	Faces	GFC~[82]/SI~[56]/CA~[21]/PEPSI~[26]/RFR-Net~[28]/EdgeConnect~[22]/StructureFlow~[78] CSA~[38]/MEDFE~[87]/DF-Net~[101]/ISK~[86]/RN~[130]/GL~[24]/GMCNN~[104] MUSICAL~[39]/PII-GAN~[50]JPG-Net~[27]
CelebA-HQ	30,000	High-resolution images	Faces	CA~[21]/PEN-Net~[36]/PIC-Net~[17]/HiFill~[32]/DSI-VQVAE~[52]/GatedConv~[47]/PartialConv~[23] VC-Net~[35]/BAT~[42]/GMCNN~[104]
Helen	2,330	Images/Parsing	Faces	GFC~[82]/SPG-Net~[76]/Colla-GAN~[83]
DTD	5,640	Images	Textures	CA~[21]/PEN-Net~[36]
FFHQ	7,000	High-resolution images	Faces	ICT~[94]/VC-Net~[35]
Facade	606	Images	Facades	PEN-Net~[36]/ISK~[86]
Cityscapes	25,000	Images/Segmentation	Scenes	SGE-Net~[81]/SPG-Net~[76]
SVHN	99,289	Images	Digitals	SI~[56]/PG-GAN~[58]
Stanford Cars	16,185	Images	Cars	SI~[56]/PG-GAN~[58]
Celeb-ID	100,000	Images	Faces	GFC~[82]
SUNCG	45,622	3D scenes	Rooms	DQ-Net~[9]
NVIDIA Irregular Mask	67,116	Binary images	Masks	PIC-Net~[17]/HiFill~[32]/EdgeConnect~[22]/LBAM~[48]/PartialConv~[23]/MEDFE~[87]/SGE-Net~[81] ISK~[86]/RN~[130]
Foreground-aware	110,000	Binary images	Masks	Foreground~[34]/ProFill~[30]

to measure the quality of reconstruction. On the other hand, inception score (IS) [132], Frèchet inception distance (FID) [133], the standard metrics for assessing the quality of GANs, are used to measure the quality of the generated image samples. User study, different from the evaluation metrics above, is a subjective evaluation metric. In this section, we briefly introduce the working principle for each evaluation metric.

MSE measures the average of the squares of errors, that is, the average squared difference between the network prediction $\mathtt{I}_{out}$ and the ground truth image $\mathtt{I}_{gt}$:

$$\mathtt{MSE} = \frac{1}{C_j \times H_j \times W_j} \| \mathtt{I}_{gt} - \mathtt{I}_{out} \|_2^2 \tag{11}$$

where $\mathtt{I}_{out}$ and $\mathtt{I}_{gt}$ are of size $C_j \times H_j \times W_j$.

In practice, mean normal squared error (NMSE) is used to make a comparison:

$$\mathtt{NMSE} = \frac{\| \mathtt{I}_{gt} - \mathtt{I}_{out} \|_2^2}{\| \mathtt{I}_{gt} \|_2^2} \tag{12}$$

PSNR, the evaluation metric widely used in image compression, measures the quality of reconstruction. Based on MSE, PSNR is defined as:

$$\mathtt{PSNR} = 10 \log_{10} \left(\frac{\mathtt{MAX}^2}{\mathtt{MSE}} \right) \tag{13}$$

where $\mathtt{MAX}$ is the maximum possible pixel value of images (e.g. 255).

SSIM measures the similarity between two images. Unlike MSE and PSNR that estimate absolute errors, SSIM perceives the change in structural information. It is based on three comparison measurements between x and y samples: (1) luminance l; (2) contrast c; and (3) structure s. SSIM is the weighted combination:

$$\mathtt{SSIM} = [l(x,y)]^{\alpha} [c(x,y)]^{\beta} [s(x,y)]^{\gamma} \tag{14}$$

LPIPS [131] compares the perceptual similarity of two images based on deep features. We formulate it as:

$$\mathtt{LPIPS} = \sum_l \frac{1}{H_l \times W_l} \left\| w_l \odot \left(\Phi_{gt}^l - \Phi_{out}^l \right) \right\|_2^2 \tag{15}$$

where Φ_{gt}^l and Φ_{out}^l are the feature maps of the l_{th} layer of the pretrained network.

IS [132] applies the pre-trained Inception-v3 network [134] to the generated samples $\hat{x}$ to obtain the corresponding labels and compares the conditional label distribution with the marginal label distribution:

$$\mathtt{IS} = \exp \left(\mathbb{E}_{\hat{x} \sim \mathbb{P}_g} D_{KL} (p(y|x) \| p(y)) \right) \tag{16}$$

On the one hand, generating images with meaningful objects leads to the conditional label distribution $p(y|x)$ with low entropy; on the other hand, generating images with diverse objects results in the marginal label distribution $p(y)$ with high entropy. Therefore, based on the KL divergence, higher IS is better.

FID [133] uses the Frèchet distance to compare the statistics of the generated samples with the real samples:

$$\mathtt{FID} = \| \mu_r - \mu_g \|_2^2 + \mathtt{Tr} \left(\sigma_r + \sigma_g - 2(\sigma_r \sigma_g)^{\frac{1}{2}} \right) \tag{17}$$

where $\mathcal{X}_r \sim \mathcal{N}(\mu_r, \sigma_r)$ and $\mathcal{X}_g \sim \mathcal{N}(\mu_g, \sigma_g)$ are the activations of layer pool-3 of the pre-trained Inception-v3 network [134] for the real and generated samples, respectively. Intuitively, lower FID is better, since the distance between the distributions of the real and generated samples are closer.

(a) Original image (b) Mask (c) Sketch map (d) Color map

Fig. 11. Examples of the inputs [135].

(a) Identity ✔ (b) Identity ✔ (c) Identity ✖

Fig. 12. Examples of the performance against face recognizers [80].

User study can be divided into single image and multiple images user study [34]. For the single image user study, a image sampled from real data or generated from a certain algorithm is shown to the volunteers to guess whether the image is real or not. For the multiple images user study, multiple images generated from different algorithms are shown to volunteers. The volunteers are required to rank them or select the most realistic result. Sometimes, only two algorithms are compared to prove the superiority of the proposed algorithm.

Discussion. Various evaluation metrics evaluate the inpainting results from different perspective. Lower MSE shows that the results are closer to the ground truth images. Lower PSNR demonstrates that the results have less noise. SSIM evaluates the similarity in structural information. LPIPS evaluates whether the results are consistent with the human perception. Higher IS means the results contain meaningful and diverse objects. Lower FID implies the distribution of the results are closer to that of the real samples. User study evaluates the results in a subjective way.

7. Applications

Compared to traditional inpainting algorithms, deep learning-based algorithms have demonstrated the superiority as the missing regions get more complicated. Through continuous improvements, deep learning-based algorithms have played an important role in user-guided face editing, privacy protection, pose-guided image synthesis, digitization of cultural heritage, remote sensing, and other fields.

7.1. Image inpainting in user-guided face editing

User-guided face editing, as the name describes, aims to edit faces by providing user inputs as guidance. In practice, unwanted elements are masked and replaced with user-drawn strokes. Portenier et al. [135] and Jo et al. [136] propose a face editing system that takes a mask, a sketch map and a color map as input to generate new faces. Fig. 11 illustrates the inputs.

7.2. Image inpainting in privacy protection

As more and more personal photos are shared online, obfuscating identities or removing objects in such photos is becoming necessary for privacy protection [80]. For obfuscating identities, Sun et al. [80] propose a head inpainting obfuscation technique. It performs effectively against face recognizers, as it is shown in Fig. 12. Interestingly, Ma et al. [137] do a reverse work that inpaints face images with masks to improve the performance of face verification. For removing objects, Upenik et al. [138] propose an object removal technique in a reversible manner. Only those who possess the private decryption key have the access to the hidden data.

(a) Original image (b) Self-occlusion UV map (c) Inpainted UV map

Fig. 13. Examples of the UV map completion [140].

(a) (b) (c)

Fig. 14. Examples of the corrupted Dunhuang murals [5].

7.3. Image inpainting in pose-Guided image synthesis

Learning human appearance from a single image has recently become an area of high research interest [139]. Specifically, the task can be described as resynthesizing the view of faces or bodies from new viewpoints. Deng et al. [140] proposes a UV completion network to generate synthetic faces with arbitrary poses. As it is shown in Fig. 13, the UV map is inpainted and the completed UV map, together with the corresponding 3D model, is used to synthesize 2D face images. Grigorev et al. [139] not only resynthesize the view of faces, but also focus on human bodies, including faces and garments.

7.4. Image inpainting in digitization of cultural heritage

Rare culture relics and sites have been on the verge of great danger because of the natural disaster, economic and tour development [141]. It is urgent and important to protect and exploit the cultural heritages. Take murals for example. Many paintings have been seriously damaged for hundreds and thousands of years. Fig. 14 illustrates some examples. Applying inpainting algorithms to murals helps the digitization and conservation of the cultural heritages.

Existing murals are not only corrupted, but also small in scale. Wen et al. [142] train a network with general datasets. Experimental results show that the network can still perform well in murals. Chen et al. [143] generate enough training data by a sliding window method in the augmentation process, and the original size murals can be divided into many small patches. For the stable training, Cao et al. [144] propose an enhanced consistent GAN model, not only improving the generalization ability, but also speeding up the computation.


(a) Landsat ETM+ SLC-off


(b) Cloud cover


(c) Shadow cover

Fig. 15. Examples of the corrupted remote sensing images [148].

7.5. Image inpainting in remote sensing

Inspired by the rapid development of satellite technology, remote sensing images have been widely used in various applications, including scene recognition [145], object detection [146], and land-use classification [147]. However, due to the bad working conditions of satellite sensors and weather conditions, remote sensing images often suffer from dead pixels, cloud covers, and shadow covers [148]. Even though a remote sensing image is clear, it may also suffer from the scene occlusions caused by redundant objects. Removing the covers and restoring the original images are crucial for the subsequent image processing and application.

Shao et al. [148] propose to solve three typical reconstruction tasks: (1) Landsat ETM+ SLC-off; (2) cloud removal; and (3) shadow removal. Fig. 15 illustrates the specific tasks. The network is pretrained on ImageNet [117], and fine-tuned by remote sensing images. Following the enhanced consistent GAN model, Xu et al. [147] uses reconstruction loss to generate straight edges with reasonable structure, and uses adversarial loss to generate complex edges based on the results of reconstruction loss.

8. Performance

Most papers have done qualitative and quantitative evaluations. However, the algorithms to be compared and the evaluation metrics to be used are quite distinct. It is not reasonable to collect results from different papers for the comparison, as the testing sets in each paper are also different. To make a fair comparison, we use the same testing sets on the inpainting algorithms whose source codes are open and pretrained models are released, 19 in total.

In this section, we first introduce our designed experiments for testing the inpainting algorithms listed in Table 1. Then we use different evaluation metrics to indicate the effectiveness of these algorithms. Finally, we discuss the performance from four aspects, datasets, mask sizes, image sizes and model efficiency.

8.1. Implementation details

We test inpainting algorithms on five datasets of ImageNet [117], Paris StreetView [118], Places2 [119], CelebA [120], and CelebA-HQ [121] with the holes of different types and hole-to-image area ratios and evaluate them by NMSE, PSNR, SSIM, LPIPS, IS, and FID. In ImageNet, the testing set contain 200 images. In Paris StreetView, the testing set contain 100 images. In Places2, CelebA, and CelebA-HQ, 2000 images in the original testing sets are randomly selected to correspond with the NVIDIA Irregular Mask dataset which covers different hole-to-image area ratios: $(0.01, 0.1]$, $(0.1, 0.2]$, $(0.2, 0.3]$, $(0.3, 0.4]$, $(0.4, 0.5]$, $(0.5, 0.6]$. Since most inpainting algorithms are trained with images of size 256×256, resizing is required. For Places2 and CelebA, the shape of which are not square, so images of size 256×256 are cropped and scaled from the full-resolution. For Paris StreetView and CelebA-HQ, images are scaled to 256×256 directly.

We list the performance evaluation in Appendix B. Here, we compare the performance from the aspects of datasets, mask sizes, image sizes, and model efficiency.

8.2. Datasets

For datasets, the inpainting results on Paris StreetView, CelebA, and CelebA-HQ are better than those on ImageNet and Places2. Especially for CelebA and CelebA-HQ, although the inpainting results are not the same as the original images, there is no doubt that the generated samples are able to mix the spurious with the genuine. But for ImageNet and Places2, the inpainting algorithms only perform well in object removal cases. As it is shown in Fig. 16, the mask is corresponding to the shape of the woman, so the inpainting result is visually realistic. If only a part of the woman is masked, it is impossible for the algorithms to restore the woman. Thinking in reverse, the woman can be detected by the difference between the original image and the inpainting result. Based on the property, inpainting algorithms can also be used in anomaly detection [149]. The content of datasets raises another question, overfitting. Most inpainting algorithms are trained with the image size of 256×256, therefore, resizing the original images of each dataset is necessary. For Places2, the process of resizing is not rigorous, cropping and scaling or just scaling, even not resizing. But for CelebA and CelebA-HQ, it demonstrates a large difference. Intuitively, the pretrained model should adapt to any face images, otherwise it has no application value. However, the process of resizing the training and testing sets must be consistent, otherwise the inpainting results will appear to be visually unrealistic. Fig. 17 illustrates examples of the failure.

8.3. Mask sizes

According to the performance evaluation[1], we find that nearly all the inpainting algorithms perform well when the hole-to-image area ratios are small, e.g., 10–20%. Therefore, a more in-depth discussion is made about the situation where the hole-to-image area ratios are large, e.g., 50–60%. Considering the test results on ImageNet and CelebA-HQ are relatively few, we make the comparison on Paris StreetView, Places2, and CelebA. We summarize the top three results evaluated by each evaluation metric and the corresponding algorithms, as listed in Table 3. Among them, RFR-Net [28], EdgeConnect [22], LBAM [48], and DF-Net [101] always keep better quantitative results. We also show the qualitative results in Fig. 18–20. In Fig. 18, the input image contains only the corner of the window, but RFR-Net [28] recovers the entire window, while LBAM [48] and EdgeConnect [22] not. In Fig. 7, the results of LBAM [48] and DF-Net [101] are visually better than that of EdgeConnect [22]. In Fig. 19, only DF-Net [101] generates visually realistic results, while RFR-Net [28] and EdgeConnect [22] recover the eye region poorly. That is to say, each algorithm has its strength on different datasets. Except for the above algorithms that generate a single result, we also discuss pluralistic inpainting that generates more than one visually plausible results. When the hole-to-image area ratios are large, general VAE-based pluralistic inpainting, such as PIC-Net [17], may fail to recover the missing regions, while the transformer module can be employed to improve the results. Fig. 21 shows the comparison.

8.4. Image sizes

In addition to the influence of hole-to-image area ratios on different algorithms, another question is about image sizes. Initially limited by computational memory, low-to-high-resolution inpainting had been proposed to ameliorate the problem. With the advancement of computer performance, the inpainting algorithms can now be directly trained with images of size 512×512, and

[1] The detailed results can be found in Appendix B.

(a) Original image (b) Mask (c) Inpainting result

Fig. 16. Case of object removal.

(a) Original size (b) Cropping and scaling (c) Scaling only (d) Scaling and cropping

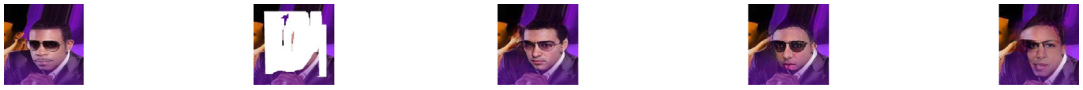
Fig. 17. (Left to Right) Original images, input images, and inpainting results.

(a) Original image (b) Input image (c) RFR-Net [28] (d) LBAM [48] (e) EdgeConnect [22]

Fig. 18. Quantitative results on Paris StreetView with irregular masks: 50–60%.

(a) Original image (b) Input image (c) LBAM [48] (d) DF-Net [101] (e) EdgeConnect [22]

Fig. 19. Quantitative results on Places2 with irregular masks: 50–60%.



(a) Original image (b) Input image (c) DF-Net [101] (d) RFR-Net [28] (e) EdgeConnect [22]

Fig. 20. Quantitative results on CelebA with irregular masks: 50–60%.

(a) Original image, input image, PIC-Net [17], and ICT [41] (b) Original image, input image, PIC-Net [17], and BAT [42]

Fig. 21. Comparison of PIC-Net [17], ICT [41] and BAT [42].

even larger [101]. For much larger sizes, such as 4096×4096, nearly all the models cause the Out-Of-Memory (OOM) error [32]. Yi et al. [32] propose the CRA mechanism to solve the problem. The mechanism is untrainable and easy to implement in other models.

8.5. Model efficiency

For model efficiency, it presents challenges to inpainting tasks, especially for real-time applications. We summarize the model sizes and testing times of the inpainting algorithms in our experiments, as listed in Table 4. The models are implemented on TensorFlow v2.5.0, PyTorch v1.12.0, CUDA v11.3.1, CUDNN v8.2.1 and run on the hardware with one CPU Intel(R) Core(TM) i7-9700KF (3.60GHz) and one GPU RTX 3060. Most of the algorithms process images on the time scale of milliseconds, and the model sizes do not seem to have significant influence. As a plug-and-play framework, JPG-Net[27] just takes a few extra milliseconds when combines with the original deep generative networks. For those algorithms processing images on the time scale of seconds, even hundred of seconds, they use time-consuming modules. For example, DSI-VQVAE [52] employs autoregressive models, and ICT [41] and BAT [42] exploit transformers [43]. DGP [25] spends the longest time, which is the inherent limitation of GAN-inversion structure.

9. Challenges and future directions

In the past decade, great progress has been made in the inpainting-strategy selection, network-structure design, loss-function optimization, etc. However, there are still many challenges in this field.

Algorithms: The current inpainting algorithms achieve better results in processing simple scenes, small holes, and low-resolution images [65]. Otherwise, these algorithms can hardly obtain satisfactory results. For example, it is still difficult to predict the structure of a missing object. Here, additional information would be required to assist the inpainting algorithm, e.g., line structure of the missing object. In addition, it is often ineffective to train an inpainting model with small datasets. Here, few-shot learning is a potential solution.

Datasets: Most datasets for inpainting simulate the missing regions of images from existing datasets such as ImageNet, Paris StreetView, Places2, CelebA, and CelebA-HQ. The simulation varies greatly across different researchers, which leads to an unfair com-

Table 3
Quantitative results on Paris StreetView, Places2, and CelebA with irregular masks: 50–60%. ↓ Lower is better. ↑ Higher is better.

Datasets	NMSE↓		PSNR↑		SSIM↑		LPIPS↓		IS↑		FID↓	
	Algorithms	Values	Algorithms	Values	Algorithms	Values	Algorithms	Values	Algorithms	Values	Algorithms	Values
Paris StreetView	RFR-Net~[28]	14.08%	RFR-Net~[28]	22.10	RFR-Net~[28]	0.734	RFR-Net~[28]	0.2042	EdgeConnect~[22]	3.03	RFR-Net~[28]	66.0
	LBAM~[48]	14.86%	LBAM~[48]	21.47	LBAM~[48]	0.712	LBAM~[48]	0.2326	LBAM~[48]	2.92	EdgeConnect~[22]	81.6
	MEDFE~[87]	15.92%	EdgeConnect~[22]	21.39	EdgeConnect~[22]	0.712	EdgeConnect~[22]	0.2380	RFR-Net~[28]	2.88	LBAM~[48]	85.6
Places2	LBAM~[48]	17.05%	LBAM~[48]	19.07	DF-Net~[101]	0.741	LBAM~[48]	0.2447	DF-Net~[101]	12.41	EdgeConnect~[22]	36.7
	DF-Net~[101]	17.94%	EdgeConnect~[22]	18.86	LBAM~[48]	0.738	EdgeConnect~[22]	0.2559	JPG-Net~[27]	10.81	LBAM~[48]	36.8
	EdgeConnect~[22]	18.20%	JPG-Net~[27]	18.81	EdgeConnect~[22]	0.731	JPG-Net~[27]	0.2571	EdgeConnect~[22]	10.76	JPG-Net~[27]	37.0
CelebA	DF-Net~[101]	14.17%	DF-Net~[101]	22.45	DF-Net~[101]	0.875	DF-Net~[101]	0.1149	DF-Net~[101]	3.55	DF-Net~[101]	12.2
	RFR-Net~[28]	14.27%	RFR-Net~[28]	21.61	RFR-Net~[28]	0.868	RFR-Net~[28]	0.1366	RFR-Net~[28]	3.48	RFR-Net~[28]	14.6
	EdgeConnect~[22]	15.36%	EdgeConnect~[22]	21.52	EdgeConnect~[22]	0.864	EdgeConnect~[22]	0.1438	EdgeConnect~[22]	2.85	EdgeConnect~[22]	14.7

Table 4
Model efficiency of different backbones.

Backbones	CE~[1]	CA~[21]	PEN-Net~[36]	PIC-Net~[17]	RFR-Net~[28]	HiFill~[32]	DSI-VQVAE~[52]	EdgeConnect~[22]	GatedConv~[47]	LBAM~[48]
Model Sizes	5.1M	3.6M	10.2M	9.2M	31.2M	2.7M	76.2M	67.1M	4.0M	68.3M
Testing Times	5.8ms	19.0ms	43.8ms	145.2ms	27.9ms	1.1s	90.3s	15.9ms	23.5ms	10.3ms
Backbones	ICT~[41]	DGP~[25]	MEDFE~[87]	DF-Net~[101]	BAT~[42]	RN~[130]	GL~[24]	GMCNN~[104]	JPG-Net~[27]	
Model Sizes	111.3M	93.5M	130.3M	32.8M	101.7M	14.3M	5.8M	12.4M	N/A	
Testing Times	31.3s	270.7s	74.6ms	7.9ms	36.8s	22.1ms	35.2ms	36.7ms	+10.9ms	

parison on the results or inpainting models. It is meaningful to build protocols for the datasets, or to collect an inpainting dataset with real damages.

Evaluation metrics: There is still a lack of unified standards to evaluate inpainting algorithms [71]. Some researches use PSNR, while others use SSIM. In our opinion, the metrics of MSE, PSNR, SSIM, and LPIPS would be a good combination for the quantitative evaluation. If GANs are involved, IS and FID would be employed as the additional metrics. Meanwhile, we think a subjective user study is a necessary metric for the qualitative evaluation. How to build a protocol for user study in the context of inpainting still requires investigation.

Applications: In most applications, users are required to accurately annotate the boundaries of the damage regions, which is time-consuming and labor-intensive. For example, when using inpainting algorithms to remove a person in a video, we first should annotate the boundary of the person frame by frame. However, it is still a challenge to segment partially-occluded objects in complex scenes. Meanwhile, it is difficult to adapt an inpainting model from one data modality to another, e.g., from nature scene images to mural paintings. At this point, domain adaptation would be a good tool to narrow the gap between the features in different domains.

10. Conclusions

This survey presented a comprehensive review about recent advances in deep learning-based image inpainting. Others in the field could benefit from the survey: (1) Develop new inpainting algorithms based on common-used inpainting strategies, network structures, and loss functions. (2) Acquire representative public datasets to conduct experiments. (3) Understand the challenges and future directions in the field, and try to achieve breakthrough research results.

In this survey, we concluded that: (1) For inpainting strategies, progressive inpainting, structural information-guided inpainting and attention-based inpainting all relies on additional information. Specifically, attention-based inpainting considers the information from farther locations than the other two strategies. However, all these three strategies may confront with the error-accumulation problem. Compared with the above three strategies, convolutions-aware inpainting can achieve better results through special convolutions when dealing with irregular holes, and pluralistic inpainting can generate various results. (2) For network structures, autoencoder-based structure and variational autoencoder-based structure both learn the latent representation of a given image, and GAN-inversion structure trains a fixed GAN generator to find a latent vector. Generally, autoencoder-based structure and GAN-inversion structure generate a single result, while variational autoencoder-based structure may generate diverse results. (3) For loss functions, generally, reconstruction loss is used to restore the overall structure, and adversarial loss is used to generate appearance close to realistic images. There are also some special loss functions, e.g., perceptual loss, MRF loss, and TV loss, which can be used as additional constrains to improve the visual quality.

Based on our own quantitative and qualitative evaluation, we found that: (1) For datasets, the inpainting algorithms are task-oriented, where the performance of a specific inpainting model would decrease dramatically when doing an inpainting task across the datasets, e.g., models trained on CelebA or CelebA-HQ perform badly for test images from Places2. (2) For mask sizes, the performance of inpainting algorithms generally gets worse with the increase of the mask size. When the mask size is large, RFR-Net, EdgeConnect, LBAM, and DF-Net always keep better quantitative results, but each has its strength on different datasets. For pluralistic inpainting, ICT and BAT, which employ the transformer module, demonstrate the superiority with the mask of large size. (3) For model efficiency, testing time is generally independent of model sizes. GAN-inversion structure and other modules, such as autoregressive models and transformers, will quadratically increase the computational cost.

Compared to other related work, we had the following advantages: (1) We provided a structured overview of deep learning-based image inpainting from multiple perspectives. (2) We discussed the advantages and disadvantages of each type of algorithms, respectively. (3) We tested pretrained models with the same testing sets to make a fair comparison of these algorithms. However, some drawbacks still existed: (1) We discussed more about the commonalities of each class, with some representative features being ignored. (2) The application scenarios of image inpainting are quite extensive, but we summarized only five of them. (3) Because the pretrained models do not provide the details of the number of epochs, batch size, and training data, the algorithm comparison is still not entirely fair.

In the future, the following work still required investigation: (1) To effectively train an inpainting model with small datasets, the future work may refer to few-shot learning. (2) To make a fair comparison on the results of inpainting models, building protocols for the datasets is necessary. (3) User study in the context of inpainting also requires protocol building. (4) To generalize the application scenarios of an inpainting model, domain adaptation should be considered.

Declaration of Competing 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.

Acknowledgements

This research was supported by the National Key Research and Development Program of China under grant 2020YFC1522703, and the National Natural Science Foundation of China under grants 62171324, 61872277.

Appendix A. Summary Table

Table A.1

Table A.1
Summary of the representative algorithms, descriptions, advantages, and disadvantages from multiple perspectives.

Multiple Perspectives	Algorithms	Description & Advantages	Description & Disadvantages
Inpainting Strategies			
Progressive Inpainting			
Coarse-to-Fine Inpainting	CA~[21]	Attention module: Propagate contextual information	Two-stage network: Increase computational cost
	PEPSI~[26]	Single-stage network and parallel decoder: Reduce computational cost Attention module: Propagate contextual information	N/A
	JPG-Net~[27]	Plug-and-play framework: Reduce computational cost Be easy to implement	N/A
Part-to-Full Inpainting	PGN~[29]	LSTM: Link stage-wise information	Four-stage network: Increase computational cost Accumulate errors
	ProFill~[30]	Confidence map: Eliminate errors	Iterative network: Increase computational cost
	RFR-Net~[28]	Feature reasoning module: Reduce computational cost	Iterative network: Accumulate errors
Low-to-High-Resolution Inpainting	NPS~[31]	Three-stage network: Adapt to high-resolution images	Three-stage network: Increase computational cost Limit to image size of 512×512
	HiFill~[32]	Two-stage network: Adapt to high-resolution images Attention module: Reduce computational cost Adapt to image size of 8000×8000	N/A
Structure-to-Content Inpainting	E-CE~[33] EdgeConnect~[22] Foreground~[34]	Edge information: Be easy to extract	Edge information: Generate blurred texture Be quite different from content
	SPG-Net~[76]	Segmentation information: Generate clear boundary	Segmentation information: Differ between appearance and semantic labels
	StructureFlow~[78]	Edge-preserved smooth information: Preserve edges and color Be similar to content	N/A
	HeadInpainting~[80]	Landmark information: Preserve facial components	N/A
Mask-to-Image Inpainting	VC-Net~[35]	Two-stage network and mask predication network: Be label-free	N/A
Structural Information-Guided Inpainting			
Edge-Guided Inpainting	GatedConv~[47]	User-guided framework and edge information: Be easy to interact	Edge information: Generate blurred texture Ignore high-frequency texture
Segmentation-Guided Inpainting	SGE-Net~[81]	Segmentation information: Generate clear boundary Deep learning support: (Theoretically) Extract segmentation information accurately	Deep learning support: (Practically) Fail to extract from damaged images
	GFC~[82]	Segmentation information: Generate clear boundary Face parsing: Preserve facial components	N/A
Landmark-Guided Inpainting	Colla-GAN~[83] DE-GAN~[84]	Collaborative network: Stabilize training Landmark information: Preserve facial components	Landmark information: Fail to extract from profiles
Voxel-Guided Inpainting	DQ-Net ~[9]	Voxel information: Inpaint depth maps	N/A
Gradient-Guided Inpainting	ISK~[86]	Gradient information: Be similar to content	N/A

(*continued on next page*)

Table A.1 (*continued*)

Multiple Perspectives	Algorithms	Description & Advantages	Description & Disadvantages
Attention-Based Inpainting			
Contextual Attention-Based Inpainting	CA~[21]	Attention module: Propagate contextual information Cosine-based similarity: Be easy to implement	Cosine-based similarity: Ignore feature magnitude Patch-wise similarity: Limit to regular holes images
	PEPSI~[26]	Attention module: Propagate contextual information Euclidean distance-based similarity: Consider feature angles and magnitude	Patch-wise similarity: Limit to regular holes images
	MA~[40]	Attention module: Propagate contextual information Pixel-wise similarity: Adapt to irregular holes images	N/A
Attention Transfer-Based Inpainting	PEN-Net~[36] HiFill~[32] ProFill~[30]	Attention module: Guide cross-scale images	N/A
Cross Attention-Based Inpainting	UCT-GAN~[18]	Attention module: Constrain instance images	N/A
Patch Swap-Based Inpainting	IMT~[37]	Attention module: Propagate contextual information	Single patch swap: Lack information
	CSA~[38]	Attention module: Propagate contextual information Similar and neighbour patches swap: Preserve global structure Preserve local consistency	N/A
	MUSICAL~[39]	Attention module: Propagate contextual information Multi-scale patches swap: Preserve global consistency Preserve local details	N/A
Transformer-Based Inpainting	ICT~[41] BAT~[42]	Attention module: Generate pluralistic results	Attention module: Increase computational cost
Convolutions-Aware Inpainting			
Partial Convolutions-Based Inpainting	PartialConv~[23]	Partial convolutional layers: Adapt to irregular holes images Be easy to implement	Partial convolutional layers: Update with hard rules Disappear progressively Lack flexibility
Gated Convolutions-Based Inpainting	GatedConv~[47]	Gated convolutional layers: Adapt to irregular holes images Pixel-wise mask learning: Be flexible	Pixel-wise mask learning: Increase computational cost
	HiFill~[32]	Gated convolutional layers: Adapt to irregular holes images Simplified mask learning: Be flexible Reduce computational cost	N/A
Bidirectional Convolutions-Based Inpainting	LBAM~[48]	Bidirectional convolutions layers: Adapt to irregular holes images Pixel-wise mask learning: Be flexible Forward and reverse mask learning: Consider decoding stages	Pixel-wise mask learning: Increase computational cost
Region-Wise Convolutions-Based Inpainting	RN~[49]	Region-wise convolutional layers: Adapt to irregular holes images Known region convolutional layers: Reconstruct original information Missing region convolutional layers: Propagate contextual information	N/A
Pluralistic Inpainting			
GAN-Based Pluralistic Inpainting	PII-GAN~[50] PD-GAN~[51]	Random vectors: Generate pluralistic results	Random vectors: Be unstable to train
VAE-Based Pluralistic Inpainting	PIC-Net~[17] UCT-GAN~[18]	Variational autoencoder: Generate pluralistic results Ground truth or instance image Stabilize training	Variational autoencoder: Be unstable to train
	DSI-VQVAE~[52]	VQ-VAE and autoregressive model: Generate pluralistic results Stabilize training	Autoregressive model: Increase computational cost

(*continued on next page*)

Table A.1 (*continued*)

Multiple Perspectives	Algorithms	Description & Advantages	Description & Disadvantages
Transformer-Based Pluralistic Inpainting	ICT~[41] BAT~[42]	Transformer module: Generate pluralistic results	Transformer module: Increase computational cost
Network structures			
Autoencoder-Based Structure			
CNN-Based Structure	CE~[1]	Channel-wise fully connected layers: Propagate contextual information Reduce computational cost	Channel-wise fully connected layers: Accept fixed image size
	NPS~[31]	Channel-wise fully connected layers: Propagate contextual information Reduce computational cost ELU activation layers: Stabilize training	
	SI~[96]	Channel-wise fully connected layers: Propagate contextual information Reduce computational cost Upsampling layers and convolutional layers Avoid checkerboard artifacts	
FCN-Based Structure	GL~[24]	Fully convolutional network: Accept arbitrary image size Dilated convolutional layers: Propagate contextual information	N/A
	EdgeConnect~[22]	Fully convolutional network: Accept arbitrary image size	
	SPG-Net~[76]	Dilated convolutional layers and residual blocks: Propagate contextual information Stabilize training	
U-Net-Based Structure		Fully convolutional network and skip links: Accept arbitrary image size Link encoder and decoder information	N/A
	PartialConv~[23]		
	DI-Net~[93]	Fully convolutional network and skip links: Accept arbitrary image size Link encoder and decoder information Network in network: Preserve global structure Preserve local details Preserve invariant features	
	DF-Net~[101]	Fully convolutional network and skip links: Accept arbitrary image size Link encoder and decoder information Fusion blocks: Preserve global consistency	
Variational Autoencoder-Based Structure			
Variational Autoencoder-Based Structure	PIC-Net~[17] UCT-GAN~[18]	Variational autoencoder: Generate pluralistic results Ground truth or instance image Stabilize training	Variational autoencoder: Be unstable to train
	DSI-VQVAE~[52]	VQ-VAE and autoregressive model: Generate pluralistic results Stabilize training	
GAN-Inversion Structure			
GAN-Inversion Structure	SI~[56]	GAN-inversion: Be flexible Multi applications	GAN-inversion: Be unstable to train Increase computational cost Differ between natural images and actual images
	PG-GAN~[58]	GAN-inversion: Be flexible Multi applications Latent vector learning: Reduce computational cost	

(*continued on next page*)

Table A.1 (*continued*)

Multiple Perspectives	Algorithms	Description & Advantages	Description & Disadvantages
	DGP~[25]	GAN-inversion: Be flexible Multi applications Fine-tuning stage: Fill gaps between natural images and actual images	
Loss functions			
Reconstruction Loss			
Weighted Reconstruction Loss	SI~[56]	Reconstruction loss: Capture overall structure Pixel-wise weighted terms: Constrain more on boundary regions	Pixel-wise weighted terms: Increase computational cost
	PG-GAN~[58]		
	GMCNN~[104]		
	HiFill~[32]	Reconstruction loss: Capture overall structure Region-wise weighted terms: Constrain more on boundary regions Reduce computational cost	N/A
Multi-Scale Reconstruction Loss	SGE-Net~[81]	Reconstruction loss: Capture overall structure Multi-scale loss: Guide decoding stages progressively	Multi-scale loss: Increase computational cost
	PEN-Net~[36]		
	DF-Net~[101]		
Adversarial Loss			
WGAN-Based Adversarial Loss	CA~[21]	Adversarial loss: Generate realistic results Gradient penalty term: Stabilize training	GAN: Be unstable to train
LSGAN-Based Adversarial Loss	PIC~[17]	Adversarial loss: Generate realistic results Least-squares: Stabilize training	GAN: Be unstable to train
	MEDFE~[87]		
	CSA~[38]		
Global and Local Adversarial Loss	GL~[24]	Adversarial loss: Generate realistic results Global discriminator: Preserve global coherence Local discriminator: Preserve local consistency	GAN: Be unstable to train Local discriminator: Fail to handle dispersed holes
PatchGAN-Based Adversarial Loss	GatedConv~[47]	Adversarial loss: Generate realistic results Patch discriminator: Preserve global coherence Preserve local consistency Spectral normalization Stabilize training	GAN: Be unstable to train
	PEPSI~[26]		
Others			
Perceptual Loss	PartialConv~[23]	VGG-based feature reconstruction loss: Generate visually indistinguishable results Preserve overall content and structure VGG-based style reconstruction loss Consider color, texture, and exact shapes	N/A
	SPG-Net~[76]	Discriminator-based feature reconstruction loss: Extract features from structure maps	
	EdgeConnect~[22]		
	Ex-GAN~[113]	Projection-based feature reconstruction loss: Match results with instance images	
	UCT-GAN~[18]		
MRF Loss	NPS~[31]	Euclidean distance-based nearest neighbour search: Preserve local texture	Euclidean distance-based nearest neighbour search: Increase computational cost
	GMCNN~[104]	Relative distance-based nearest neighbour search: Preserve local texture Diversify structure	Relative distance-based nearest neighbour search: Increase computational cost
TV Loss	PartialConv~[23]	Total variation regularization Preserve spatial smoothness	N/A

Appendix B. Performance Evaluation

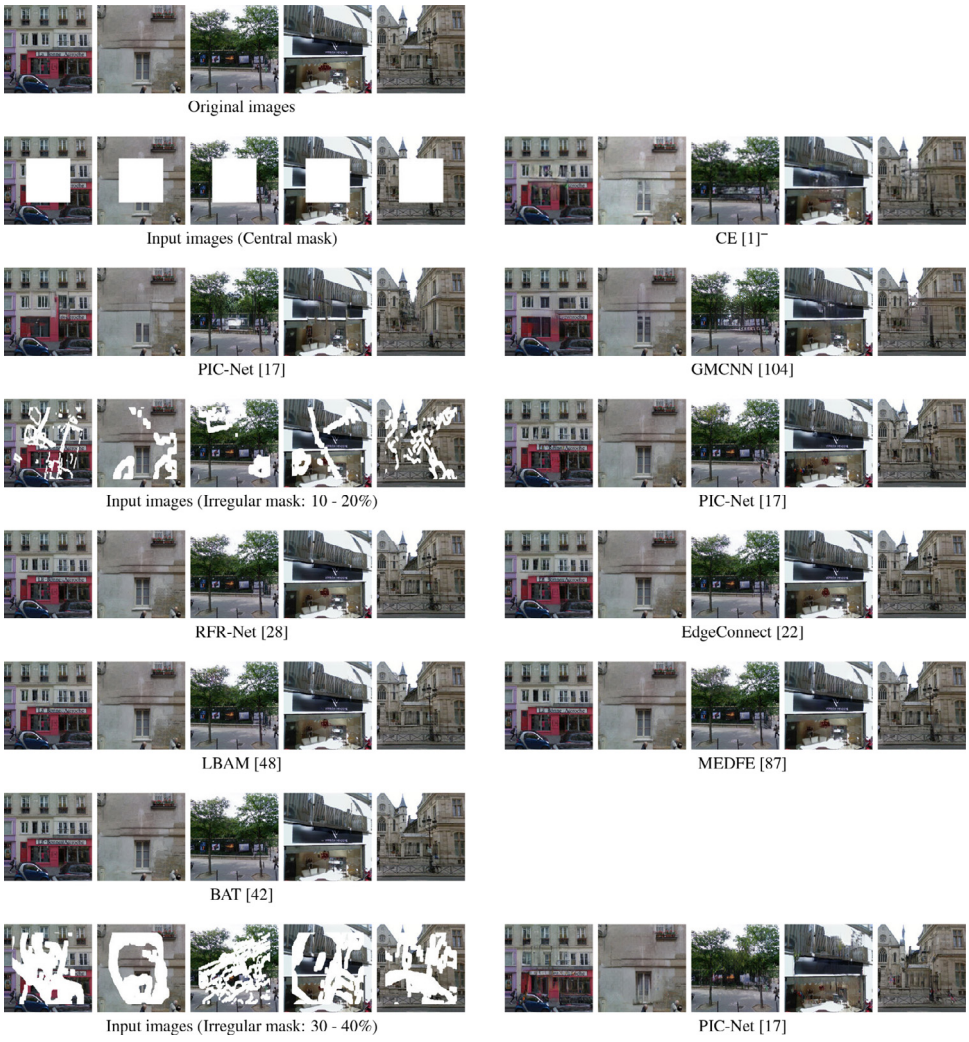
Tables B.1–B.6

Table B.1
Quantitative results on five datasets with different models in our experiments. ↓ Lower is better. ↑ Higher is better.

Algorithms	Mask Types	Mask Ratios	Image Sizes	Imagenet						Paris StreetView						Places2						CelebA						CelebA-HQ					
				NMSE↓	PSNR↑	SSIM↑	LPIPS↓	IS↑	FID↓	NMSE↓	PSNR↑	SSIM↑	LPIPS↓	IS↑	FID↓	NMSE↓	PSNR↑	SSIM↑	LPIPS↓	IS↑	FID↓	NMSE↓	PSNR↑	SSIM↑	LPIPS↓	IS↑	FID↓	NMSE↓	PSNR↑	SSIM↑	LPIPS↓	IS↑	FID↓
CE~[1]	Central(64 × 64)		128 × 128	9.30%	22.93	0.895	0.1220	11.50	57.2	6.65%	22.11	0.856	0.1061	2.64	37.8																		
CA~[21]	Central(128 × 128)		256 × 256	8.99%	21.34	0.872	0.1743	12.36	66.0							9.22%	20.24	0.854	0.1547	12.83	24.2	8.09%	25.29	0.950	0.0728	3.52	6.6	8.06%	24.14	0.947	0.0739	3.06	8.7
PEN-Net~[36]	Central(128 × 128)		256 × 256													10.35%	20.92	0.857	0.2128	10.55	41.0							8.28%	25.51	0.954	0.0695	2.98	10.2
PIC-Net~[17]	Central(128 × 128)		256 × 256	10.36%	20.27	0.859	0.1924	11.60	80.5	7.36%	23.61	0.850	0.1242	3.02	44.9	10.42%	19.61	0.845	0.1649	12.28	26.0							7.78%	24.37	0.947	0.0618	3.11	5.6
	Irregular(10–20%)		256 × 256	5.28%	27.08	0.933	0.1052	13.92	37.3	4.53%	28.36	0.918	0.0708	2.93	32.7	5.74%	26.33	0.925	0.0790	15.46	13.5							3.96%	30.21	0.966	0.0372	3.17	5.0
	Irregular(30–40%)		256 × 256	12.60%	21.51	0.833	0.2313	11.59	105.5	10.74%	23.41	0.800	0.1725	3.13	66.2	13.51%	21.21	0.818	0.1914	11.83	29.5							9.48%	24.60	0.916	0.0954	2.93	10.5
	Irregular(50–60%)		256 × 256	19.37%	16.00	0.659	0.4327	5.86	224.5	20.87%	18.62	0.620	0.3372	2.68	102.7	22.68%	17.06	0.671	0.3475	7.73	52.4							17.07%	19.49	0.828	0.1924	2.70	19.8
RFR-Net~[28]	Irregular(10–20%)		256 × 256							2.72%	31.68	0.951	0.0367	2.99	15.6							2.29%	33.32	0.980	0.0207	3.87	3.3						
	Irregular(30–40%)		256 × 256							7.16%	26.38	0.869	0.0954	3.01	35.9							6.61%	26.95	0.941	0.0607	3.72	8.0						
	Irregular(50–60%)		256 × 256							14.08%	22.10	0.734	0.2042	2.88	66.0							14.27%	21.61	0.868	0.1366	3.48	14.6						
HiFill~[32]	Irregular(10–20%)		512 × 512													6.04%	24.53	0.912	0.1118	17.30	66.0												
	Irregular(30–40%)		512 × 512													13.34%	19.42	0.787	0.2487	6.57	127.1												
	Irregular(50–60%)		512 × 512													21.52%	15.55	0.613	0.4244	4.36	210.7												
DSI-VQVAE~[52]	Central(128 × 128)		256 × 256	11.51%	19.56	0.842	0.1894									9.94%	19.65	0.833	0.1530									7.42%	25.80	0.949	0.0574		
EdgeConnect~[22]	Irregular(10–20%)		256 × 256							3.49%	30.28	0.939	0.0469	3.02	21.3	4.68%	27.49	0.941	0.0523	16.42	9.0	2.94%	32.42	0.977	0.0245	3.54	3.5						
	Irregular(30–40%)		256 × 256							8.61%	25.41	0.849	0.1160	3.14	45.3	10.69%	22.56	0.852	0.1340	13.79	20.3	7.78%	26.55	0.937	0.0663	3.29	8.0						
	Irregular(50–60%)		256 × 256							15.98%	21.39	0.712	0.2380	3.03	81.6	18.20%	18.86	0.731	0.2559	10.76	36.7	15.36%	21.52	0.864	0.1438	2.85	14.7						
GatedConv~[47]	Central(128 × 128)		256 × 256													9.96%	19.82	0.854	0.1545	13.70	24.1							7.93%	25.36	0.951	0.0620	3.03	7.0
LBAM~[48]	Irregular(10–20%)		256 × 256							2.75%	31.44	0.949	0.0384	2.99	17.1	3.64%	28.67	0.951	0.0424	16.13	7.2												
	Irregular(30–40%)		256 × 256							7.40%	25.98	0.860	0.1073	3.05	43.3	9.31%	23.11	0.864	0.1210	14.07	18.6												
	Irregular(50–60%)		256 × 256							14.86%	21.47	0.712	0.2326	2.92	85.6	17.05%	19.07	0.738	0.2447	10.41	36.8												
ICT~[41]	Irregular(10–20%)		256 × 256	6.37%	24.53	0.928	0.0797									7.23%	25.48	0.926	0.0523														
	Irregular(30–40%)		256 × 256	13.77%	19.85	0.819	0.1788									14.39%	20.99	0.809	0.1451														
	Irregular(50–60%)		256 × 256	20.16%	16.52	0.695	0.3075									24.21%	16.86	0.668	0.2773														
DGP~[25]	Central(128 × 128)		256 × 256	29.07%	17.74	0.643	0.2885																										
MEDFE~[87]	Irregular(10–20%)		256 × 256							2.78%	32.37	0.957	0.0358	2.91	18.1	4.26%	28.75	0.951	0.0521	15.57	10.1												
	Irregular(30–40%)		256 × 256							7.52%	26.18	0.870	0.1065	3.14	44.0	10.70%	22.76	0.859	0.1535	11.95	29.3												
	Irregular(50–60%)		256 × 256							15.92%	20.80	0.703	0.2456	2.81	87.3	18.92%	18.42	0.716	0.3040	6.81	60.6												
DF-Net~[101]	Irregular(10–20%)		256 × 256 512 × 512													3.91% (512)	28.17 (512)	0.948 (512)	0.0566 (512)	18.49 (512)	7.9 (512)	2.36% (256)	34.24 (256)	0.981 (256)	0.0173 (256)	3.91 (256)	2.9 (256)						
	Irregular(30–40%)		256 × 256 512 × 512													9.98% (512)	22.74 (512)	0.862 (512)	0.1456 (512)	16.42 (512)	20.0 (512)	6.82% (256)	27.86 (256)	0.945 (256)	0.0505 (256)	3.76 (256)	6.9 (256)						
	Irregular(50–60%)		256 × 256 512 × 512													17.94% (512)	18.70 (512)	0.741 (512)	0.2780 (512)	12.41 (512)	39.9 (512)	14.17% (256)	22.45 (256)	0.875 (256)	0.1149 (256)	3.55 (256)	12.2 (256)						
BAT~[42]	Irregular(10–20%)		256 × 256							3.27%	28.51	0.945	0.0335			4.27%	27.22	0.947	0.0345									2.05%	34.25	0.986	0.0148		
	Irregular(30–40%)		256 × 256							9.64%	22.67	0.841	0.1037			10.94%	21.74	0.839	0.1131									7.80%	25.71	0.936	0.0630		
	Irregular(50–60%)		256 × 256							17.28%	17.56	0.675	0.2506			20.69%	17.40	0.688	0.2651									13.77%	20.74	0.873	0.1411		
RN~[130]	Irregular(10–20%)		256 × 256													5.11%	28.29	0.948	0.0659	15.79	10.8												
	Irregular(30–40%)		256 × 256													12.48%	23.08	0.866	0.1490	13.67	23.4												
	Irregular(50–60%)		256 × 256													22.02%	18.72	0.723	0.3028	8.41	54.3												
GL~[24]	Central(128 × 128)		256 × 256													9.37%	20.96	0.855	0.1581	12.47	25.5												
GMCNN~[104]	Central(128 × 128)		256 × 256							7.27%	24.38	0.859	0.1243	2.90	45.9													5.80%	25.88	0.957	0.0526	3.09	6.1
JPG-Net~[27]	Irregular(10–20%)		256 × 256													4.72%	27.37	0.940	0.0532	16.26	9.0	3.02%	32.25	0.976	0.0251	3.54	3.6						
	Irregular(30–40%)		256 × 256													10.75%	22.47	0.850	0.1356	13.90	20.5	7.92%	26.42	0.936	0.0673	3.27	8.1						
	Irregular(50–60%)		256 × 256													18.24%	18.81	0.730	0.2571	10.81	37.0	15.46%	21.46	0.864	0.1446	2.84	14.8						

Table B.2
Qualitative results on ImageNet with different models in our experiments. $^{+}$ Size 512 × 512. $^{-}$ Size 128 × 128.

Table B.3
Qualitative results on Paris StreetView with different models in our experiments. $^{+}$ Size 512 × 512. $^{-}$ Size 128 × 128.



(*continued on next page*)

Table B.3 (*continued*)

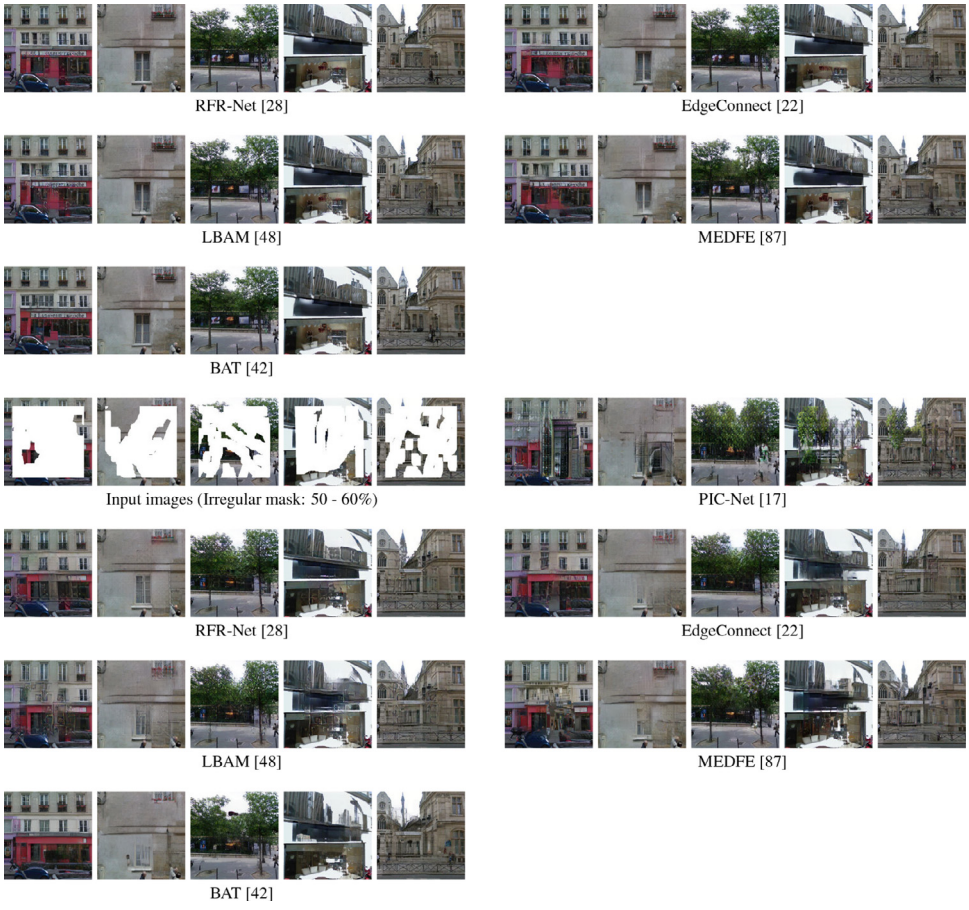
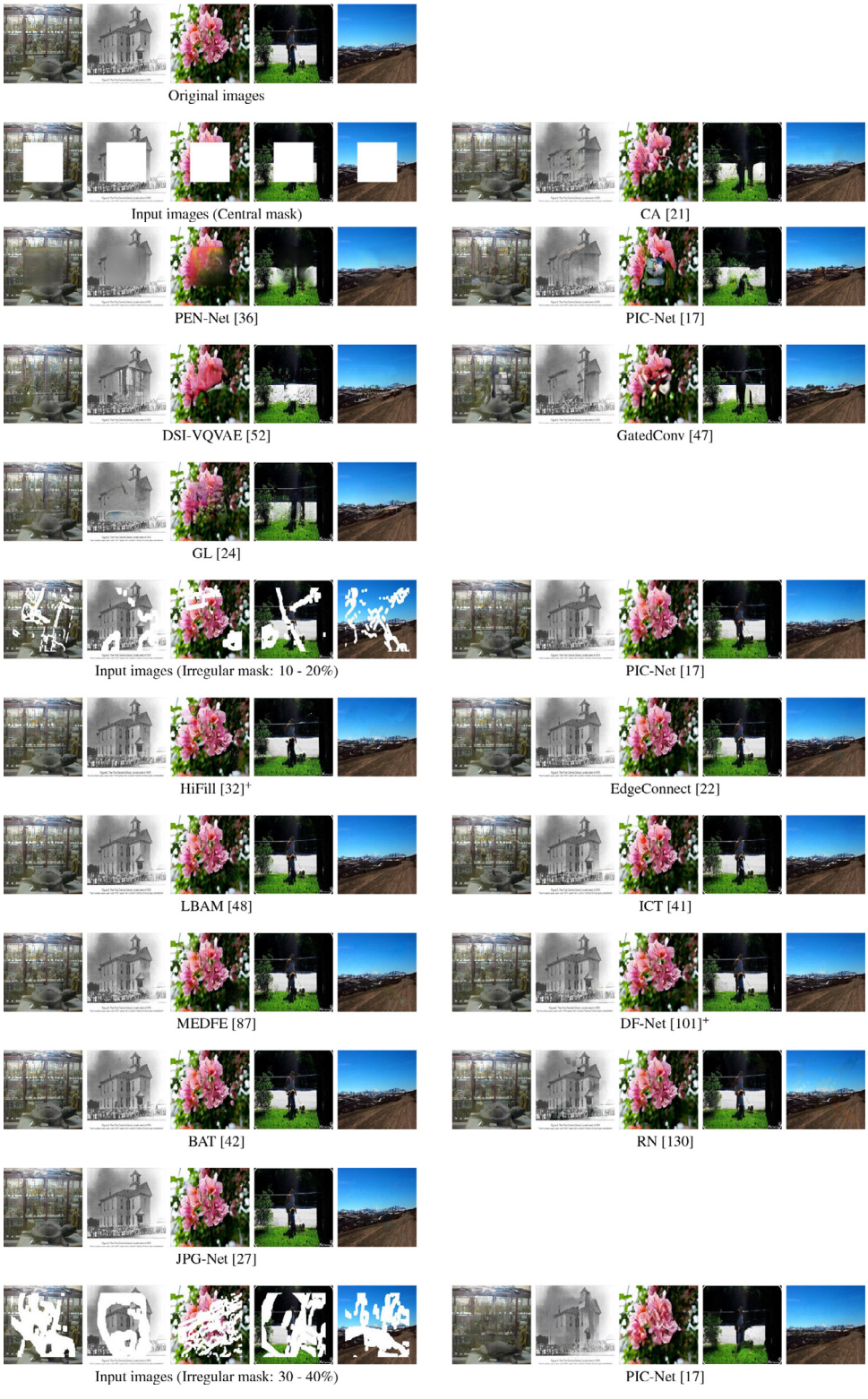


Table B.4
Qualitative results on Places2 with different models in our experiments. $^{+}$ Size 512 × 512. $^{-}$ Size 128 × 128.



(*continued on next page*)

Table B.4 (*continued*)

Table B.5
Qualitative results on CelebA with different models in our experiments. $^{+}$ Size 512 × 512. $^{-}$ Size 128 × 128.

Table B.6
Qualitative results on Paris CelebA-HQ with different models in our experiments. $^{+}$ Size 512 × 512. $^{-}$ Size 128 × 128.

References

[1] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A.A. Efros, Context encoders: Feature learning by inpainting, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2536–2544.
[2] T.F. Chan, J. Shen, Nontexture inpainting by curvature-driven diffusions, J. Vis. Commun. Image Represent 12 (4) (2001) 436–449.
[3] M. Bertalmio, L. Vese, G. Sapiro, S. Osher, Simultaneous structure and texture image inpainting, IEEE Trans. Image Process. 12 (8) (2003) 882–889.
[4] K. Sasaki, S. Iizuka, E. Simo-Serra, H. Ishikawa, Learning to restore deteriorated line drawing, Vis. Comput. 34 (2018) 1077–1085.
[5] H. Wang, Q. Li, Q. Zou, Inpainting of dunhuang murals by sparsely modeling the texture similarity and structure continuity, J. Comput. Cult. Heritage (JOCCH) 12 (3) (2019) 1–21.
[6] S. Al-Takrouri, A.V. Savkin, A model validation approach to texture recognition and inpainting, Pattern Recognit. 43 (6) (2010) 2054–2067.
[7] L. Zhang, A.M. Yip, M.S. Brown, C.L. Tan, A unified framework for document restoration using inpainting and shape-from-shading, Pattern Recognit. 42 (11) (2009) 2961–2978.
[8] Q. Zou, Y. Cao, Q. Li, Q. Mao, S. Wang, Automatic inpainting by removing fence-like structures in RGBD images, Mach. Vis. Appl. 25 (7) (2014) 1841–1858.
[9] X. Han, Z. Zhang, D. Du, M. Yang, J. Yu, P. Pan, X. Yang, L. Liu, Z. Xiong, S. Cui, Deep reinforcement learning of volume-guided progressive view inpainting for 3D point scene completion from a single depth image, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 234–243.
[10] Z. Pei, M. Jin, Y. Zhang, M. Ma, Y.-H. Yang, All-in-focus synthetic aperture imaging using generative adversarial network-based semantic inpainting, Pattern Recognit. 111 (2021) 107669.
[11] A. Criminisi, P. Pérez, K. Toyama, Region filling and object removal by exemplar-based image inpainting, IEEE Trans. Image Process. 13 (9) (2004) 1200–1212.
[12] C. Barnes, E. Shechtman, A. Finkelstein, D.B. Goldman, PatchMatch: a randomized correspondence algorithm for structural image editing, ACM Trans. Graph. 28 (3) (2009) 24–33.
[13] D. Ding, S. Ram, J.J. Rodriguez, Perceptually aware image inpainting, Pattern Recognit. 83 (2018) 174–184.
[14] M. Bertalmio, G. Sapiro, V. Caselles, C. Ballester, Image inpainting, in: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 2000, pp. 417–424.
[15] J. Ji, B. Zhong, K.-K. Ma, Image interpolation using multi-scale attention-aware inception network, IEEE Trans. Image Process. 29 (2020) 9413–9428.
[16] L. Yu, K. Liu, M.T. Orchard, Manifold-inspired single image interpolation, arXiv preprint arXiv:2108.00145(2021).
[17] C. Zheng, T.-J. Cham, J. Cai, Pluralistic image completion, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1438–1447.
[18] L. Zhao, Q. Mo, S. Lin, Z. Wang, Z. Zuo, H. Chen, W. Xing, D. Lu, UCTGAN: diverse image inpainting based on unsupervised cross-space translation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 5740–5749.
[19] J. Xie, L. Xu, E. Chen, Image denoising and inpainting with deep neural networks, Advances in Neural Information Processing Systems (NIPS), vol. 25, 2012.
[20] A. Fawzi, H. Samulowitz, D. Turaga, P. Frossard, Image inpainting through neural networks hallucinations, in: Image, Video, and Multidimensional Signal Processing Workshop (IVMSP), 2016, pp. 1–5.
[21] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, T.S. Huang, Generative image inpainting with contextual attention, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 5505–5514.
[22] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, M. Ebrahimi, EdgeConnect: Structure guided image inpainting using edge prediction, in: IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019, pp. 3265–3274.
[23] G. Liu, F.A. Reda, K.J. Shih, T.-C. Wang, A. Tao, B. Catanzaro, Image inpainting for irregular holes using partial convolutions, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 85–100.
[24] S. Iizuka, E. Simo-Serra, H. Ishikawa, Globally and locally consistent image completion, ACM Trans. Graph. 36 (4) (2017) 1–14.
[25] X. Pan, X. Zhan, B. Dai, D. Lin, C.C. Loy, P. Luo, Exploiting deep generative prior for versatile image restoration and manipulation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2020, pp. 262–277.
[26] M.-c. Sagong, Y.-g. Shin, S.-w. Kim, S. Park, S.-j. Ko, PEPSI: fast image inpainting with parallel decoding network, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 11360–11368.
[27] Q. Guo, X. Li, F. Juefei-Xu, H. Yu, Y. Liu, S. Wang, JPGNet: joint predictive filtering and generative network for image inpainting, in: Proceedings of the 29th ACM International Conference on Multimedia, 2021, pp. 386–394.
[28] J. Li, N. Wang, L. Zhang, B. Du, D. Tao, Recurrent feature reasoning for image inpainting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7757–7765.
[29] H. Zhang, Z. Hu, C. Luo, W. Zuo, M. Wang, Semantic image inpainting with progressive generative networks, in: Proceedings of the 26th ACM International Conference on Multimedia (ACM MM), 2018, pp. 1939–1947.
[30] Y. Zeng, Z. Lin, J. Yang, J. Zhang, E. Shechtman, H. Lu, High-resolution image inpainting with iterative confidence feedback and guided upsampling, in: Proceedings of the European Conference on Computer Vision (ECCV), 2020, pp. 1–17.
[31] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, H. Li, High-resolution image inpainting using multi-scale neural patch synthesis, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4076–4084.
[32] Z. Yi, Q. Tang, S. Azizi, D. Jang, Z. Xu, Contextual residual aggregation for ultra high-resolution image inpainting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7508–7517.
[33] L. Liao, R. Hu, J. Xiao, Z. Wang, Edge-aware context encoder for image inpainting, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 3156–3160.
[34] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes, J. Luo, Foreground-aware image inpainting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5833–5841.
[35] Y. Wang, Y.-C. Chen, X. Tao, J. Jia, VCNet: a robust approach to blind image inpainting, in: Proceedings of the European Conference on Computer Vision (ECCV), 2020, pp. 752–768.
[36] Y. Zeng, J. Fu, H. Chao, B. Guo, Learning pyramid-context encoder network for high-quality image inpainting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1486–1494.
[37] Y. Song, C. Yang, Z. Lin, X. Liu, Q. Huang, H. Li, C.-C.J. Kuo, Contextual-based image inpainting: infer, match, and translate, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 3–19.
[38] H. Liu, B. Jiang, Y. Xiao, C. Yang, Coherent semantic attention for image inpainting, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 4169–4178.
[39] N. Wang, J. Li, L. Zhang, B. Du, Musical: multi-scale image contextual attention learning for inpainting, in: IJCAI, 2019, pp. 3748–3754.
[40] N. Wang, S. Ma, J. Li, Y. Zhang, L. Zhang, Multistage attention network for image inpainting, Pattern Recognit. 106 (2020) 107448.
[41] Z. Wan, J. Zhang, D. Chen, J. Liao, High-fidelity pluralistic image completion with transformers, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 4692–4701.
[42] Y. Yu, F. Zhan, R. Wu, J. Pan, K. Cui, S. Lu, F. Ma, X. Xie, C. Miao, Diverse image inpainting with bidirectional and autoregressive transformers, in: Proceedings of the 29th ACM International Conference on Multimedia (ACM MM), 2021, pp. 69–78.
[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in Neural Information Processing Systems (NIPS), vol. 30, 2017.
[44] R. Köhler, C. Schuler, B. Schölkopf, S. Harmeling, Mask-specific inpainting with deep neural networks, in: German Conference on Pattern Recognition, 2014, pp. 523–534.
[45] J.S. Ren, L. Xu, Q. Yan, W. Sun, Shepard convolutional neural networks, Advances in Neural Information Processing Systems (NIPS), volume 28, 2015.
[46] A. Dapogny, M. Cord, P. Pérez, The missing data encoder: Cross-channel image completion with hide-and-seek adversarial network, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), vol. 34, 2020, pp. 10688–10695.
[47] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, T.S. Huang, Free-form image inpainting with gated convolution, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 4470–4479.
[48] C. Xie, S. Liu, C. Li, M.-M. Cheng, W. Zuo, X. Liu, S. Wen, E. Ding, Image inpainting with learnable bidirectional attention maps, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 8857–8866.
[49] Y. Ma, X. Liu, S. Bai, L. Wang, D. He, A. Liu, Coarse-to-fine image inpainting via region-wise convolutions and non-local correlation, in: IJCAI, 2019, pp. 3123–3129.
[50] W. Cai, Z. Wei, PiiGAN: generative adversarial networks for pluralistic image inpainting, IEEE Access 8 (1) (2020) 48451–48463.
[51] H. Liu, Z. Wan, W. Huang, Y. Song, X. Han, J. Liao, PD-GAN: probabilistic diverse GAN for image inpainting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 9371–9381.
[52] J. Peng, D. Liu, S. Xu, H. Li, Generating diverse structure for image inpainting with hierarchical VQ-VAE, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 10775–10784.
[53] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[54] O. Ronneberger, P. Fischer, T. Brox, U-Net: convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015, pp. 234–241.
[55] P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, Image-to-image translation with conditional adversarial networks, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5967–5976.
[56] R.A. Yeh, C. Chen, T. Yian Lim, A.G. Schwing, M. Hasegawa-Johnson, M.N. Do, Semantic image inpainting with deep generative models, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6882–6890.
[57] P. Vitoria, J. Sintes, C. Ballester, Semantic image inpainting through improved wasserstein generative adversarial networks, arXiv preprint arXiv: 1812.01071(2018).
[58] A. Lahiri, A.K. Jain, S. Agrawal, P. Mitra, P.K. Biswas, Prior guided GAN based semantic inpainting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 13693–13702.
[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems (NIPS), vol. 27, 2014, pp. 2672–2680.

[60] L. Wang, W. Chen, W. Yang, F. Bi, F.R. Yu, A state-of-the-art review on image synthesis with generative adversarial networks, IEEE Access 8 (2020) 63514–63537.

[61] S. Su, M. Yang, L. He, X. Shao, Y. Zuo, Z. Qiang, A survey of face image inpainting based on deep learning, in: International Conference on Cloud Computing, 2022, pp. 72–87.

[62] M.H. Yap, N. Batool, C.-C. Ng, M. Rogers, K. Walker, A survey on facial wrinkles detection and inpainting: datasets, methods, and challenges, IEEE Trans. Emerg. Top.Comput. Intell. 5 (4) (2021) 505–519.

[63] H.-y. Zhang, Q.-c. Peng, A survey on digital image inpainting, J. Image Graph. 12 (1) (2007) 1–10.

[64] B.H. Patil, P. Patil, A comprehensive review on state-of-the-art image inpainting techniques, Scalable Comput. Pract. Exp. 21 (2) (2020) 265–276.

[65] Z. Qin, Q. Zeng, Y. Zong, F. Xu, Image inpainting based on deep learning: a review, Displays (2021) 102028.

[66] Y. Weng, S. Ding, T. Zhou, A survey on improved GAN based image inpainting, in: 2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE), 2022, pp. 319–322.

[67] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, Y. Akbari, Image inpainting: a review, Neural Process. Lett. 51 (2) (2020) 2007–2028.

[68] T. Haofeng, D. Yuanfang, Z. Yitong, S. Juanjuan, Survey of image inpainting algorithms based on deep learning, Comput. Sci. 47 (2) (2020) 161–174.

[69] Z. Qiang, L. He, X. Chen, D. Xu, Survey on deep learning image inpainting methods, J. Image Graph. 24 (3) (2019) 447–463.

[70] B. Coloma, B. Aurelie, H. Samuel, P. Simone, V. Patricia, An analysis of generative methods for multiple image inpainting, arXiv preprint arXiv:2205.02146(2022).

[71] J. Jam, C. Kendrick, K. Walker, V. Drouard, J.G.-S. Hsu, M.H. Yap, A comprehensive review of past and present image inpainting methods, Comput. Vis. Image Understanding (2020) 103147.

[72] N.M.F. Salem PhD, A survey on various image inpainting techniques, Future Eng. J. 2 (2) (2021) 1.

[73] S. Mehra, From textural inpainting to deep generative models: an extensive survey of image inpainting techniques, Int. J. Trends Comput.Sci. (2) (2020).

[74] A. Mohammed, S. Yildirim, I. Farup, M. Pedersen, Ø. Hovde, Y-Net: a deep convolutional neural network for polyp detection, arXiv preprint arXiv:1806.01907(2018).

[75] H. Samet, M. Tamminen, Efficient component labeling of images of arbitrary dimension represented by linear bintrees, IEEE Trans Pattern Anal Mach Intell 10 (4) (1988) 579–586.

[76] Y. Song, C. Yang, Y. Shen, P. Wang, Q. Huang, C.-C. J. Kuo, SPG-Net: segmentation prediction and guidance network for image inpainting, arXiv preprint arXiv:1805.03356(2018).

[77] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 801–818.

[78] Y. Ren, X. Yu, R. Zhang, T.H. Li, S. Liu, G. Li, Structureflow: image inpainting via structure-aware appearance flow, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 181–190.

[79] L. Xu, Q. Yan, Y. Xia, J. Jia, Structure extraction from texture via relative total, ACM Trans. Graph. 31 (6) (2012) 1–10.

[80] Q. Sun, L. Ma, S.J. Oh, L. Van Gool, B. Schiele, M. Fritz, Natural and effective obfuscation by head inpainting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (ICCV), 2018, pp. 5050–5059.

[81] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, S. Satoh, Guidance and evaluation: semantic-aware image inpainting for mixed scenes, in: Proceedings of the European Conference on Computer Vision (ECCV), 2020, pp. 683–700.

[82] Y. Li, S. Liu, J. Yang, M.-H. Yang, Generative face completion, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5892–5900.

[83] H. Liao, G. Funka-Lea, Y. Zheng, J. Luo, S.K. Zhou, Face completion with semantic knowledge and collaborative adversarial learning, in: Asian Conference on Computer Vision (ACCV), 2018, pp. 382–397.

[84] X. Zhang, X. Wang, C. Shi, Z. Yan, X. Li, B. Kong, S. Lyu, B. Zhu, J. Lv, Y. Yin, et al., DE-GAN: domain embedded GAN for high quality face image inpainting, Pattern Recognit. (2021) 108415.

[85] S. Song, F. Yu, A. Zeng, A.X. Chang, M. Savva, T. Funkhouser, Semantic scene completion from a single depth image, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 190–198.

[86] J. Yang, Z. Qi, Y. Shi, Learning to incorporate structure knowledge for image inpainting, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2020, pp. 12605–12612.

[87] H. Liu, B. Jiang, Y. Song, W. Huang, C. Yang, Rethinking image inpainting via a mutual encoder-decoder with feature equalizations, in: Proceedings of the European Conference on Computer Vision (ECCV), 2020, pp. 725–741.

[88] X. Wang, R. Girshick, A. Gupta, K. He, Non-local neural networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7794–7803.

[89] A. Buades, B. Coll, J.-M. Morel, A non-local algorithm for image denoising, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, 2005, pp. 60–65.

[90] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7132–7141.

[91] T. Zhou, S. Tulsiani, W. Sun, J. Malik, A.A. Efros, View synthesis by appearance flow, in: Proceedings of the European Conference on Computer Vision (ECCV), 2016, pp. 286–301.

[92] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.

[93] Q. Xiao, G. Li, Q. Chen, Deep inception generative network for cognitive image inpainting, arXiv preprint arXiv:1812.01458(2018).

[94] A. Razavi, A. van den Oord, O. Vinyals, Generating diverse high-fidelity images with VQ-VAE-2, in: Advances in Neural Information Processing Systems (NIPS), vol. 32, 2019, pp. 14866–14876.

[95] M.A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, AIChE J. 37 (2) (1991) 233–243.

[96] H.V. Vo, N.Q. Duong, P. Pérez, Structural inpainting, in: ACM International Conference on Multimedia (ACM MM), 2018, pp. 1948–1956.

[97] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, in: International Conference on Learning Representations (ICLR), 2016.

[98] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[99] Z. Guo, Z. Chen, T. Yu, J. Chen, S. Liu, Progressive image inpainting with full-resolution residual network, in: Proceedings of the 27th ACM International Conference on Multimedia (ACM MM), 2019, pp. 2496–2504.

[100] M. Lin, Q. Chen, S. Yan, Network in network, arXiv preprint arXiv:1312.4400(2013).

[101] X. Hong, P. Xiong, R. Ji, H. Fan, Deep fusion network for image completion, in: ACM International Conference on Multimedia (ACM MM), 2019, pp. 2033–2042.

[102] Y. Zeng, Y. Gong, J. Zhang, Feature learning and patch matching for diverse image inpainting, Pattern Recognit. 119 (2021) 108036.

[103] D. Ulyanov, A. Vedaldi, V. Lempitsky, Deep image prior, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 9446–9454.

[104] Y. Wang, X. Tao, X. Qi, X. Shen, J. Jia, Image inpainting via generative multi-column convolutional neural networks, in: Advances in Neural Information Processing Systems (NIPS), 2018, pp. 329–338.

[105] A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks, arXiv preprint arXiv:1511.06434(2015).

[106] C. Villani, Optimal Transport: Old and New, vol. 338, Springer, 2009.

[107] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, Improved training of Wasserstein GANs, in: Advances in Neural Information Processing Systems (NIPS), 2017, pp. 5769–5779.

[108] X. Mao, Q. Li, H. Xie, R.Y. Lau, Z. Wang, S. Paul Smolley, Least squares generative adversarial networks, in: IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2813–2821.

[109] A. Jolicoeur-Martineau, The relativistic discriminator: a key element missing from standard GAN, in: International Conference on Learning Representations (ICLR), 2019.

[110] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, in: International Conference on Learning Representations (ICLR), 2018.

[111] J. Johnson, A. Alahi, L. Fei-Fei, Perceptual losses for real-time style transfer and super-resolution, in: Proceedings of the European Conference on Computer Vision (ECCV), 2016, pp. 694–711.

[112] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556(2014).

[113] B. Dolhansky, C.C. Ferrer, Eye in-painting with exemplar generative adversarial networks, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7902–7911.

[114] B. Yan, Q. Lin, W. Tan, S. Zhou, Assessing eye aesthetics for automatic multi-reference eye in-painting, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 13506–13514.

[115] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al., Photo-realistic single image super-resolution using a generative adversarial network, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 105–114.

[116] L.A. Gatys, A.S. Ecker, M. Bethge, A neural algorithm of artistic style, arXiv preprint arXiv:1508.06576(2015).

[117] 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, Int. J. Comput. Vis. 115 (3) (2015) 211–252.

[118] C. Doersch, S. Singh, A. Gupta, J. Sivic, A. Efros, What makes paris look like paris? ACM Trans. Graph. 31 (4) (2012) 1–9.

[119] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, A. Torralba, Places: a 10 million image database for scene recognition, IEEE Trans. Pattern Anal. Mach. Intell. 40 (6) (2018) 1452–1464.

[120] Z. Liu, P. Luo, X. Wang, X. Tang, Deep learning face attributes in the wild, in: IEEE International Conference on Computer Vision (ICCV), 2015, pp. 3730–3738.

[121] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of GANs for improved quality, stability, and variation, in: International Conference on Learning Representations (ICLR), 2018.

[122] B.M. Smith, L. Zhang, J. Brandt, Z. Lin, J. Yang, Exemplar-based face parsing, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 3484–3491.

[123] V. Le, J. Brandt, Z. Lin, L. Bourdev, T.S. Huang, Interactive facial feature localization, in: Proceedings of the European Conference on Computer Vision (ECCV), 2012, pp. 679–692.
[124] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, A. Vedaldi, Describing textures in the wild, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 3606–3613.
[125] R. Tyleček, R. Šára, Spatial pattern templates for recognition of objects with regular structure, in: German Conference on Pattern Recognition (GCPR), 2013, pp. 364–374.
[126] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3213–3223.
[127] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, Advances in Neural Information Processing Systems (NIPS), 2011.
[128] J. Krause, M. Stark, J. Deng, L. Fei-Fei, 3D object representations for fine–grained categorization, in: IEEE International Conference on Computer Vision Workshops (ICCVW), 2013, pp. 554–561.
[129] E. Agustsson, R. Timofte, NTIRE 2017 challenge on single image super-resolution: dataset and study, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 1122–1131.
[130] T. Yu, Z. Guo, X. Jin, S. Wu, Z. Chen, W. Li, Z. Zhang, S. Liu, Region normalization for image inpainting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 12733–12740.
[131] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 586–595.
[132] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training GANs, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 2234–2242.
[133] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter, GANs trained by a two time-scale update rule converge to a local Nash equilibrium, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6629–6640.
[134] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818–2826.
[135] T. Portenier, Q. Hu, A. Szabo, S.A. Bigdeli, P. Favaro, M. Zwicker, FaceShop: deep sketch-based face image editing, ACM Trans. Graph. 37 (4) (2018) 1–13.
[136] Y. Jo, J. Park, SC-FEGAN: face editing generative adversarial network with user's sketch and color, in: IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1745–1753.
[137] X. Ma, X. Zhou, H. Huang, G. Jia, Z. Chai, X. Wei, Contrastive attention network with dense field estimation for face completion, Pattern Recognit. 124 (2021) 108465.
[138] E. Upenik, P. Akyazi, M. Tuzmen, T. Ebrahimi, Inpainting in omnidirectional images for privacy protection, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019, pp. 2487–2491.
[139] A. Grigorev, A. Sevastopolsky, A. Vakhitov, V. Lempitsky, Coordinate-based texture inpainting for pose-guided human image generation, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 12127–12136.
[140] J. Deng, S. Cheng, N. Xue, Y. Zhou, S. Zafeiriou, UV-GAN: adversarial facial UV map completion for pose-invariant face recognition, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7093–7102.
[141] D.-m. LU, Y.-h. PAN, R. CHEN, Dunhuang cave virtual rebuilding and mural restoration simulating, Acta Geodaetica Et Cartographic. Sin. 31 (1) (2002) 12–16.
[142] L.-l. Wen, D. Xu, X. Zhang, W.-h. Qian, The inpainting of irregular damaged areas in ancient murals using generative model, J. Graph. 40 (5) (2019) 925–931.
[143] M. Chen, X. Zhao, D. Xu, Image inpainting for digital dunhuang murals using partial convolutions and sliding window method, in: Journal of Physics: Conference Series, vol. 1302, 2019, pp. 32–40.
[144] J. Cao, Z. Zhang, A. Zhao, H. Cui, Q. Zhang, Application of enhanced consistent generative adversarial network in mural repairing, J. Comput.-Aided Des. Comput.Graph. 32 (8) (2020) 1315–1323.
[145] Q. Zou, L. Ni, T. Zhang, Q. Wang, Deep learning based feature selection for remote sensing scene classification, IEEE Geosci. Remote Sens. Lett. 12 (11) (2015) 2321–2325.
[146] C. Xu, C. Li, Z. Cui, T. Zhang, J. Yang, Hierarchical semantic propagation for object detection in remote sensing imagery, IEEE Trans. Geosci. Remote Sens. 58 (6) (2020) 4353–4364.
[147] H. Xu, X. Tang, B. Ai, X. Gao, F. Yang, Z. Wen, Missing data reconstruction in VHR images based on progressive structure prediction and texture generation, ISPRS J. Photogramm. Remote Sens. 171 (2021) 266–277.
[148] M. Shao, C. Wang, T. Wu, D. Meng, J. Luo, Context-based multiscale unified network for missing data reconstruction in remote sensing images, IEEE Geosci. Remote Sens. Lett. (2020) 1–5.
[149] V. Zavrtanik, M. Kristan, D. Skočaj, Reconstruction by inpainting for visual anomaly detection, Pattern Recognit. 112 (2021) 107706.

Hanyu Xiang was born in 1998. He is currently pursuing the PhD degree in Photogrammetry and Remote Sensing from Wuhan University, Wuhan, China. His research interests include computer vision and digitalization of cultural heritage.

Qin Zou received the B.E. degree in information engineering and the Ph.D. degree in computer vision from Wuhan University, China, in 2004 and 2012, respectively. From 2010 to 2011, he was a visiting PhD student at the Computer Vision Lab, University of South Carolina, USA. Currently, he is an associate professor with the School of Computer Science, Wuhan University. He is a co-recipient of the National Technology Invention Award of China 2015. His research activities involve computer vision, pattern recognition, and machine learning. He is serving as an Associate Editor of IEEE Transactions on Intelligent Vehicles. He is a senior member of the IEEE.

Muhammad Ali Nawaz was born in 1991. He is currently pursuing the PhD degree in Computer Science and Technology from Wuhan University, Wuhan, China. His research interests include deep learning and computer vision.

Xianfeng Huang received the PhD degree from Wuhan University, Wuhan, China. He is currently a Full Professor with State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing. His research interests include photogrammetry, computer vision, and digitalization of cultural heritage.

Fan Zhang received the PhD degree from Wuhan University, Wuhan, China. He is currently an Associate Professor with State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing. His research interests include photogrammetry and digitalization of cultural heritage.

Hongkai Yu received the PhD in Computer Science and Engineering from University of South Carolina. He is currently an assistant professor in the Department of Electrical Engineering and Computer Science at the Cleveland State University. His research interests include computer vision, machine learning, deep Learning, artificial Intelligence and data Science.