RESEARCH ARTICLE

Progressive With Purpose: Guiding Progressive Inpainting DNNs Through Context and Structure

KANGDI SHI[1], **MUHAMMAD ALRABEIAH**[2], **(Member, IEEE),**
AND JUN CHEN[1]**, (Senior Member, IEEE)**
[1]Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada
[2]Electrical Engineering Department, King Saud University, Riyadh 11451, Saudi Arabia

Corresponding author: Jun Chen (chenjun@mcmaster.ca)

ABSTRACT The advent of deep learning in the past decade has significantly helped advance image inpainting. Although achieving promising performance, deep learning-based inpainting algorithms still struggle from the distortion caused by the fusion of structural and contextual features, which are commonly obtained from, respectively, deep and shallow layers of a convolutional encoder. Motivated by this observation, we propose a novel progressive inpainting network that maintains the structural and contextual integrity of a processed image. More specifically, inspired by the Gaussian and Laplacian pyramids, the core of the proposed network is a feature extraction module named GLE. Stacking GLE modules enables the network to extract image features from different image frequency components. This ability is important to maintain structural and contextual integrity, for high frequency components correspond to structural information while low frequency components correspond to contextual information. The proposed network utilizes the GLE features to progressively fill in missing regions in a corrupted image in an iterative manner. Our benchmarking experiments demonstrate that the proposed method achieves clear improvement in performance over many state-of-the-art inpainting algorithms.

INDEX TERMS Deep image inpainting, image pyramid.

I. INTRODUCTION

Image inpainting is the task of restoring missing patches of pixels in an image [1], [2], [7]. As the name suggests, inpainting targets filling in missing parts of an image (i.e., image holes) with contextually meaningful information so that the image can be restored to its original form. This could be a quite difficult task for machines, for it is an ill-posed inverse problem [8]. It requires not only the ability to predict what is missing, but also whether it fits within the context of the image or not. Thus, a key to attaining satisfying inpainting results is to ensure that the reconstructed pixels are consistent with the uncorrupted region and exhibit coherence in both structure and texture.

As a main remedy for restoring image quality, inpainting is of great importance nowadays, for modern societies are increasingly reliant on visual content with images as its

The associate editor coordinating the review of this manuscript and approving it for publication was Li He.

building block, from surveillance systems to autonomous vehicles, media streaming, and conference calls. Storing, displaying, and exchanging huge amounts of images make them prone to damage, one of which is missing pixels (image holes). It is thus unsurprising to see increasing research interest in inpainting within the computer vision community.

A. MOTIVATION

Many image inpainting techniques have been proposed over the past two decades. They could loosely be grouped into two major categories: traditional and modern. The main defining difference between the two categories is the use of deep learning. The traditional category of techniques could collectively be divided into two sub-categories [8]: exemplar-based and diffusion-based. The former approach [9] searches for the best matching patches from known regions and pastes them into missing regions. Such techniques have high computational costs for patch searching and generate unrealistic results due to the lack of perspective transformation. The

diffusion-based techniques, on the other hand, recreate a missing region with features from its surrounding known region. Although diffusion-based techniques are more efficient than their exemplar-based counterparts, they result in over-smoothed inpainting results because of regularization based on partial-differential equations.

The advent of deep learning in computer vision has created a surge of inpainting techniques that utilize Deep Neural Networks (DNNs). Although those techniques exhibit some overlap with the tradition diffusion-based techniques, they define the state-of-the-art in inpainting, and, therefore, merit a category of their own. Most of the early works on inpainting with deep learning, like [10] and [11], follow a two-stage approach, which firstly learns the image structure from a given edge/structure map of a corrupted image, then refines the missing region with a texture generator. However, two-stage image inpainting methods usually cause artifacts due to their limited ability to recover both structure and texture.

To deal with those artifacts, progressive inpainting techniques have been explored. They rely on the idea that not all predicted pixels in a region plagued with artifacts are defective; some are good predictions that could be utilized to improve the re-generated region and weed out the artifacts. Hence, those techniques fill in the missing holes by iterating over the image and learning from previously-predicted pixels. Examples of such techniques are the full-resolution residual network proposed by Guo et al. [12] and the iterative confidence feedback network proposed by Zheng et al. [13]. Despite the improvement they provide over two-stage techniques, the performance of progressive techniques is still prone to artifacts. This could be traced back to the inexplicit modelling of structure and texture in those techniques.

Fusing structure and texture awareness with progressive inpainting is arguably the most promising approach to overcoming visual artifacts, which shall be followed in this paper. Developing structure and texture-aware algorithms has been explored recently in Guo et al. [14]. Two different but coupled autoencoders are trained with structure and texture constraints to fill in holes in corrupted images. The results are encouraging, but the algorithm can cause distortion in deep parts of the hole due to one-stage feature fusion. This could be overcome by incorporating progressive inpainting into the learning process.

B. CONTRIBUTION

In an attempt to bring together progressive learning and fusion of texture and structure, this paper presents a Gaussian-Laplacian feature Extraction (GLE) module. The main contributions of the proposed architecture are summarized below:

- **GLE Module:** Inspired by image pyramid, we propose a GLE module to obtain features from high and low image frequency components. Those components provide texture information (low-frequency components) and structure information (high-frequency components). The GLE module leverages those multi-frequency components to learn textural and structural features.

- **Iterative Reinpainting Component:** A progressive reinpainting component is developed such that it gradually fills in the corrupted regions of an image. It utilizes features learned by the GLE modules from different frequency components to fill in the outer edge of the corrupted regions iteratively until the region is restored.

- **Benchmarking and Evaluation Experiments:** Various experiments are designed to evaluate the performance of the proposed architecture and show the benefits of the GLE module and the reinpainting component. The experiments also compare the proposed inpainting algorithm to some state-of-the-art algorithms to situate its contribution to the inpainting problem.

C. PAPER ORGANIZATION

The organization of this paper is shown as follows. Section II reviews works that are related to our method. Section III details the architecture of the proposed progressive image inpainting network. Sections IV and V illustrate the experimental setup and the experimental results. Section VI concludes this paper.

II. RELATED WORK

The proposed solution is developed on top of a rich literature of image inpainting with deep learning. To facilitate the discussion, the following three subsections will review some concepts related to the proposed solution and some relevant inpainting solutions. They should lay the necessary groundwork for the detailed description in Section III

A. VARIANTS OF TEXTURE AND STRUCTURE INPAINTING

Inpainting based on texture and structure has been attempted in various forms in the literature. The concepts of *style* and *content* have been introduced in [3], which could be viewed as derivatives of texture and structures, respectively. They are used to build a two-stage inpainting DNN. In the first, two encoders extract style and content latent information separately, and the second stage synthesize a full image from that information. Semantic segmentation masks are another alternative that helps capture structure information. They have been utilized in [4], [5], and [6] as a way to guide texture generation. In all those papers, an encoder network learns to generate a latent representation of the corrupted image that captures the structure. It does so by pushing the decoder network to recover not only the inpaintd image but also its segmentation mask. The three papers differ in the details of how to encode a corrupted image and generate a structurally consistent image, but they all match in the objective, capturing the structure to produce meaningfully inpainted image.

B. PROGRESSIVE IMAGE INPAINTING

Progressive image inpainting, as the name suggests, aims to recover images gradually by utilizing features from undamaged and recently recovered regions. Overall, algorithms

following this approach could be grouped in two broad categories: (i) contextual information-based algorithms, and (ii) structural constraints-based algorithms. Both are briefly reviewed below.

Contextual information-based algorithms rely mainly on CNN features extracted from input images to restore damaged regions. As a pioneer of the contextual features-based algorithms, Hsu et al. [15] propose using several deep convolution networks to learn progressive inpainting from multiple image scales, from low to high resolution images. Zhang et al. [16] recognize how inpainting lends itself to recurrent modelling; they propose to use several generative networks inter-connected with LSTM module, which progressively fills in the missing region of an image. More recently, Li et al. [17] extend that analogy further. They propose a recurrent feature reasoning module with knowledge-consistent attention, which can progressively enhance the details in masked regions.

Compared with their contextual information-based counterparts, the structural constraints-based algorithms take advantage of additional external structural constraints provided by edge detection algorithms. The algorithms in [11] and [18] utilize contour or edge maps as a guide for image completion. To progressively complete the image, Li et al. [20] propose a U-net that recovers the edge maps while inpainting images progressively. These approaches, collectively, seek to tackle image inpainting by introducing structural constraints, yet their performance remains limited by a lack of information for recovering deeper pixels in the missing regions.

C. GAUSSIAN AND LAPLACIAN PYRAMID

A classical approach to image inpainting is centered around the idea of building multi-scale image pyramids, in which inpainting is done progressively from one scale to another—commonly from smallest to largest scale. Those pyramids are usually called Gaussian or Laplacian pyramids based on the type of filters used to generate them. Specifically, let $\mathcal{G}$ denote the Gaussian smooth operator, $\mathbf{I}_\tau$ express the input image to the τ_{th} level of Gaussian pyramid. $\mathcal{Q}$ denotes upsampling operation, $\mathcal{D}$ denotes downsampling operation. The formulas for the output images $\mathbf{G}_\tau$, $\mathbf{F}_\tau$ of τ_{th} Gaussian pyramid and Laplacian pyramid are

$$\mathbf{G}_\tau = \mathcal{D}(\mathcal{G}(\mathbf{I}_\tau)) \qquad (1)$$

$$\mathbf{F}_\tau = \mathbf{I}_\tau - \mathcal{Q}(G_\tau) \qquad (2)$$

Inpainting algorithms using Gaussian and Laplacian pyramids could roughly be clustered into three groups: inpainting on multiple Gaussian pyramids [21], inpainting on multiple Laplacian pyramids pyramids [22], [23], and inpainting on multiple Gaussian and Laplacian [23].

The difference between the first and the second group is in how the inpainting algorithm is applied on different image pyramids. For instance, Farid et al. [21] first generate multiple Gaussian pyramids until most missing pixels are eliminated by the smoothing operation. Then, their algorithm copies and pastes the missing pixels from the small-scale image (top of the pyramid) to the large-scale images (bottom of the pyramid). In contrast, [22] utilizes Laplacian pyramid with patch search to recover missing pixels from small to large scale images in the pyramid. Because of the limitation of exemplar-based methods, both kinds of methods suffer from unrealistic inpainting results.

Benefiting from the combination of structure and texture, the third inpainting group (i.e., algorithms relying on multiple Gaussian and Laplacian pyramids) usually achieve better performance than their counterparts relying only on one of the two pyramids. However, the additional cost from inpainting both pyramids is relatively high compared to that of the former two groups.

III. PROPOSED INPAINTING ALGORITHM

Like a person solving a jigsaw puzzle, an inpainting algorithm should fill in the missing regions by gradually piecing pixels together while keeping an eye on context and structure. Progressive algorithms, as mentioned earlier, restore missing pixels gradually using undamaged and recently recovered pixels, yet they do not jointly maintain contextual and structural information. This observation fuels the work in this paper; a Deep neural Network (DNN) is designed such that it progressively inpaints with the purpose of maintaining structure and context information. Hence, it is described as being progressive with purpose.

The idea behind the proposed algorithm is to break down the inpainting task into three main stages, namely feature extraction (first stage), iterative inpainting (second stage), and enhancing and reconstruction (third stage). The first stage is aimed to extract multi-level features from the corrupted image, which mimics, to some extent, feature extraction from image pyramids used in classical inpainting algorithms such as [21] and [22]. The multi-level features capture contextual and structural information. They are fed to the iterative inpainting stage, which attempts to recover some of the missing information gradually over several iterations. Each one generates a pair of feature volumes. The pairs are passed to the enhancement and reconstruction stage to enhance the recovered information, fuse them into one feature volume, and reconstruct the complete image. The architecture of the proposed algorithm is depicted in Figure 1.

The architecture is detailed in the following four subsections. The first one presents a formal description of how progressive inpainting restores missing pixels. The following three are a deep-dive into the three stages of the proposed architecture, describing the inner workings of each stage. Finally, the last subsection presents the loss function used to train the architecture.

A. RATIONALE BEHIND THE PROPOSED ALGORITHM

Let $\mathbf{a}$ be the original image, and $\mathbf{b}$ be the corrupted image. We denote the conditional probability distribution of the original image given the corrupted image by $p_{\mathbf{A}|\mathbf{B}}$. Image

inpainting can be formulated as a maximum a posterior (MAP) estimation problem:

$$\hat{\mathbf{a}}_{map} = \arg\max_{\hat{\mathbf{a}}} p_{\mathbf{A}|\mathbf{B}}(\hat{\mathbf{a}}|\mathbf{b}). \qquad (3)$$

Let $\mathbf{m}$ be the ground truth of the corrupted region, $\mathbf{n}$ and $\mathbf{c}$ be the valid region and corrupted region of the corrupted image, then $\mathbf{a} = \mathbf{m} \cup \mathbf{n}$, $\mathbf{b} = \mathbf{c} \cup \mathbf{n}$. Note that the conditional distribution of $\mathbf{m}$ given $\mathbf{n}$, denoted by $p_{\mathbf{M}|\mathbf{N}}$, is a projected version of $p_{\mathbf{A}|\mathbf{B}}$ and can be learned from the training dataset. The MAP estimation of $\mathbf{a}$ based on $\mathbf{b}$ can be reduced to the MAP estimation of $\mathbf{m}$ based on $\mathbf{n}$:

$$\hat{\mathbf{m}}_{map} = \arg\max_{\hat{\mathbf{m}}} p_{\mathbf{M}|\mathbf{N}}(\hat{\mathbf{m}}|\mathbf{n}). \qquad (4)$$

Clearly, we have

$$\hat{\mathbf{a}}_{map} = \hat{\mathbf{m}}_{map} \cup \mathbf{n}. \qquad (5)$$

Our algorithm aims to produce an approximate version of $\hat{\mathbf{m}}_{map}$.

We divide the corrupted region into T concentric regions, and progressively recover the corrupted region in an inward manner from the 1_{th} to the T_{th} concentric region.

Let $\tilde{\mathbf{m}}_r^{(\tau)}$ denote the inpainted r_{th} concentric region at the τ_{th} step. The process proceeds as follows. At the τ_{th} step (with τ from 1 to T), we generate $\tilde{\mathbf{m}}_\tau^{(\tau)}$ based on the valid region $\mathbf{n}$ and the $\tau - 1$ inpainted concentric regions $\tilde{\mathbf{m}}_r^{(\tau-1)}(1 \le r \le \tau-1)$ from the $\tau - 1_{th}$ step,[1] then refine $\tilde{\mathbf{m}}_r^{(\tau-1)}(1 \le r \le \tau - 1)$ to $\tilde{\mathbf{m}}_r^{(\tau)}(1 \le r \le \tau - 1)$, respectively.

At the end of the T_{th} step, we collected $\tilde{\mathbf{m}}_r^{(\tau)}(1 \le \tau \le T, 1 \le r \le \tau)$ generated throughout the process and perform an enhancement. Specifically, for τ from 1 to $T - 1$, we leverage $\cup_{r=1}^{\tau-1}\tilde{\mathbf{m}}_r^{(\tau-1)}$ (which is void when $\tau = 1$) and $\cup_{r=1}^{\tau}\tilde{\mathbf{m}}_r^{(\tau+1)}$, together with the valid region $\mathbf{n}$, to enhance $\cup_{r=1}^{\tau}\hat{\mathbf{m}}_r^{(\tau)}$. More precisely, for each τ, the enhancement is carried out in two parts separately: $\cup_{r=1}^{\tau-1}\tilde{\mathbf{m}}_r^{(\tau)}$ is enhanced based on $\cup_{r=1}^{\tau-1}\tilde{\mathbf{m}}_r^{(\tau-1)}$ and $\cup_{r=1}^{\tau-1}\tilde{\mathbf{m}}_r^{(\tau+1)}$ while $\tilde{\mathbf{m}}_\tau^{(\tau)}$ is enhanced based on $\tilde{\mathbf{m}}_\tau^{(\tau+1)}$. It is also worth mentioning that in our implementation, we decompose $\mathbf{n}$ into low-level information $\mathbf{l}$ and high-level information $\mathbf{h}$ using a feature extraction netwrok. Let $\hat{\mathbf{m}}_\tau$ denote the enhanced version[2] of $\cup_{r=1}^{\tau}\tilde{\mathbf{m}}_r^{(\tau)}$ $(1 \le \tau \le T)$. Our algorithm produces $\cup_{\tau=1}^{T}\hat{\mathbf{m}}_\tau$ as an approximation of $\hat{\mathbf{m}}_{map}$.

B. FEATURE EXTRACTION STAGE

The feature extraction stage is inspired by Gaussian and Laplacian pyramids. The main component of this stage is a sequence of convolution, Gaussian smoothing, upsampling, another convolution, and subtraction. This sequence will be henceforth referred to as the GLE module. As shown in the first column of Figure 1, the input is first passed through a convolutional layer with 64 kernels, and a ReLU activation function. Next, the generated feature continue passing

---

[1] when $\tau = 1$, there is no inpainted region available, but only the valid region $\mathbf{n}$.

[2] Note that $\hat{\mathbf{m}}_T = \cup_{r=1}^{T}\tilde{\mathbf{m}}_r^{(T)}$ since no enhancement is performed when $\tau = T$.

through a convolutional layer with a number of kernels that is double the number of input channels. Each kernel has a 7×7 height and width, 2×2 stride, and 3×3 padding. It results in a reduced size feature that is then blurred using a 3×3 Gaussian kernel moving with a stride of 1 and implementing a padding of 1 to maintain the spatial dimensions fixed. The smoothed feature is passed to the next feature module and to the upsampling layer, as well. It is upsampled using nearest neighbor to recover the original input size before it is passed through the second convolutional layer. This convolution is characterized with the same hyper-parameters as those of the first one, but it has half the number of kernels recovering the same number of channels as that of the input tensor. The output feature map is produced by subtracting the original input from the feature map coming from the second convolutional layer. Let $I_{\tau-1}$ denotes the input feature maps of τ_{th} GLE module, $\mathcal{G}$ denotes the gaussian smoothing operation, Up denotes the upsample operation, Λ_{Gs} denotes the weights of the convolutional layer before the gaussian smoothing operation, and Λ_{Up} denotes the weights of convolution layer after the upsample operation. The GLE module can be expressed as

$$\mathbf{F}_\tau = \mathbf{I}_{\tau-1} - \Lambda_{\mathcal{Q}}(\mathcal{Q}(\mathcal{G}(\Lambda_{\mathcal{G}}(\mathbf{I}_{\tau-1})))), \qquad (6)$$

where $\tau \in \{1, 2, \ldots, 5\}$. $\mathbf{F}_\tau$ denotes output feature maps from τ_{th} GLE module. The $\mathbf{F}_6$ is generated right after the gaussian smooth layer of 5_{th} GLE module.

To produce various levels of features, this stage is designed to have 5 GLE modules stacked consecutively, each one feeds into the next. A corrupted image, one with missing pixels and denoted by $\mathbf{I}_{in}$ in Figure 1, a corrupted structural image $\mathbf{I}_{struc}$ used in [10] and a binary mask $\mathbf{M}_{in}$ are the input to the first module, and the output is the blurred feature volume $\mathbf{I}_1$ as well as the difference feature volume $\mathbf{F}_1$. $\mathbf{I}_1$ has half the height and width of the input image and double the number of channels, and it is passed to the next GLE module. $\mathbf{F}_1$, on the other hand, is buffered to construct the feature pyramid output that represents the output of the feature extraction stage. The pyramid is formed by stacking the difference-feature volumes generated by each GLE module, namely $\mathbf{F}_1, \ldots, \mathbf{F}_6$.

C. ITERATIVE INPAINTING STAGE

This is the second stage of the proposed solution, which is based on the concept of progressive inpainting. The main elements of this stage are partial convolution, regular convolution, and feature attention. These elements make up two parallel branches, in which features are processed iteratively. The following three subsections detail the inner workings of this stage.

1) PARTIAL CONVOLUTION

Partial convolution is a fundamental tool to fill the irregular holes in deep learning-based image inpainting and keep track of the unfilled regions of the image. To see how a partial convolution layer accomplishes this, let $\mathbf{W}_k \in \mathbb{R}^{C \times H' \times W'}$
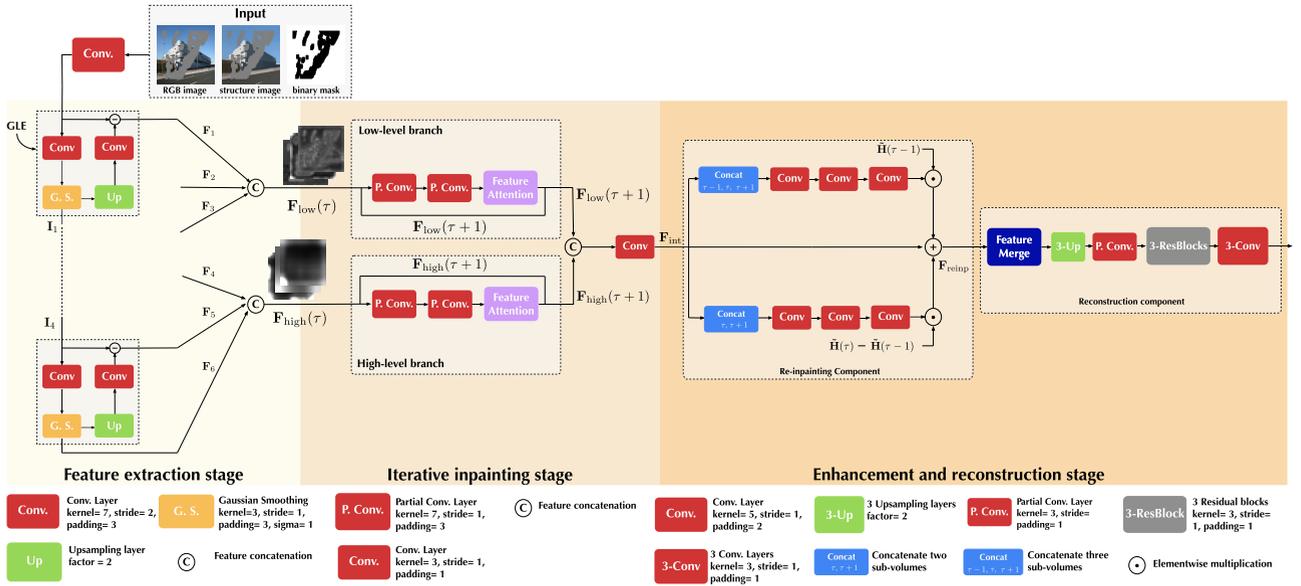
FIGURE 1. A graphical description of the proposed solution. It shows all three stages and details their main components and elements. In feature extraction stage, all input images are converted into 6 feature volumes: F_1, F_2, F_3, F_4, F_5, F_6 which are classified as low and high-frequency feature volumes: $F_{low}(0)$ and $F_{high}(0)$. In iterative inpainting stage, these feature volumes ($F_{low}(\tau)$ and $F_{high}(\tau)$ ($0 \le \tau \le 6$)) are utilized to recover the missing region progressively and generate inpainted features from each iteration, namely $F_{int}(\tau)$. Furthermore, the re-inpainting component enhances each inpainted feature by leveraging features from neighboring iterations, and it stores enhanced features as $F_{reinp}(\tau)$. In the end, the reconstruction component uses all enhanced features to produce the fully recovered image.

denote the weight tensor of the k-th kernel in a partial convolution layer, $\mathbf{X}_{i,j} \in \mathbb{R}^{C \times H' \times W'}$ denote the input feature patch extracted from the input tensor $\mathbf{X}_{in} \in \mathbf{R}^{C \times H \times W}$ centered around (i,j)-th pixels, where C, H' and W' are, respectively, the number of channels, height, and width of the patch and C, H and W are, respectively, the number of channels, height, and width of the input tensor. Also, let $\tilde{\mathbf{H}}_{i,j}$ denote a $H' \times W'$ binary patch centered around the (i,j)-th pixel, and $\mathbf{H}_{i,j}$ is a $C \times H' \times W'$ binary tensor formed by stacking C copies of the matrix $\tilde{\mathbf{H}}_{i,j}$. Then, the (i,j)-th value of the k-th output feature map, i.e., $y_{i,j,k}$, produced by a partial convolution layer—before activation—is given by

$$y_{i,j,k} = \begin{cases} g(\mathbf{W}_k, \mathbf{X}_{i,j}, \mathbf{H}_{i,j}) + b, & \sum_{C,H',W'} \mathbf{H}_{i,j} > 0, \\ 0, & \text{otherwise}, \end{cases} \quad (7)$$

where $g(\mathbf{W}_k, \mathbf{X}_{i,j}, \mathbf{H}_{i,j})$ is defined as

$$\sum_{C,H',W'} \mathbf{W}_k \odot (\mathbf{X}_{i,j} \odot \mathbf{H}_{i,j}) \frac{\sum_{C,H',W'}(\mathbf{1})}{\sum_{C,H',W'} \mathbf{H}_{i,j}}, \quad (8)$$

1 is a $C \times H' \times W'$ tensor of all ones, and $b \in \mathbb{R}$ is the bias associated with the k-th kernel. Following a partial convolution is a mask update to make sure that the mask is keeping up with the updated feature map coming out of the partial convolution. Let the full mask be given by

$$\tilde{\mathbf{H}} = \begin{bmatrix} h_{11} & \dots & h_{1W} \\ \vdots & \ddots & \vdots \\ h_{H1} & \dots & h_{HW} \end{bmatrix}, \quad (9)$$

where H and W are, respectively, the height and width of the mask such that $H \gg H'$ and $W \gg W'$, and $\tilde{\mathbf{H}}_{i,j}$ is a sub-matrix forming a block in $\tilde{\mathbf{H}}$ centered around (i,j)-th pixel. This mask is updated by convolving an all one kernel with the mask. Let $\mathbf{U}$ be a $H' \times W'$ kernel of all ones. Then, the updated mask is given by

$$\tilde{\mathbf{H}}_{new} = \tilde{\mathbf{H}}_{previous} * \mathbf{U}, \quad (10)$$

where $*$ is the convolution operation with a stride equal to that of the partial convolution kernel. More about partial convolution could be found in [24].

2) FEATURE ATTENTION

For any feature volume $\mathbf{F} \in \mathbb{R}^{C \times H \times W}$, an attention tensor could be generated using cosine similarity and softmax. Let $\mathbf{f}_{i,j}$ and $\mathbf{f}_{i',j'}$ denote pair of feature values at location i,j and i',j'. Then, their cosine similarity is computed as follows:

$$z_{i,j,i',j'} = \langle \frac{\mathbf{f}_{i,j}}{\|\mathbf{f}_{i,j}\|}, \frac{\mathbf{f}_{i',j'}}{\|\mathbf{f}_{i',j'}\|} \rangle, \quad (11)$$

where $z_{i,j,i',j'}$ denotes the cosine similarity score between the $\mathbf{f}_{i,j}$ and $\mathbf{f}_{i',j'}$.[3] Let $\mathbf{Z}_{i,j} \in \mathbb{R}^{H \times W}$ denotes the score matrix of a feature vector at location i,j and all C-dimensional feature vectors in $\mathbf{F}$. The softmax function is applied across the height and width to generate the attention score of location i,j in $\mathbf{F}$. Formally, this is expressed as follows:

$$\hat{\mathbf{Z}}_{i,j} = \text{sfm}(\mathbf{Z}_{i,j}), \quad (12)$$

[3] $\mathbf{f}_{i,j}$ is a feature vector with size $1 \times 1 \times C$ at position i,j from a feature volume with size $H \times W \times C$. $\mathbf{f}_{i,j}$ and $\mathbf{f}_{i',j'}$ have the same size, and $\|\mathbf{f}_{i,j}\|$ and $\|\mathbf{f}_{i',j'}\|$ are the second norms of those vectors.

where $\hat{\mathbf{Z}}_{i,j} \in \mathbb{R}^{H \times W}$. The final feature volume $\mathbf{F}$ has an attention tensor $\hat{\mathbf{Z}} \in \mathbb{R}^{HW \times H \times W}$ formed by stacking HW score maps $\hat{\mathbf{Z}}_{i,j}$. Based on the calculated score map, we reuse the feature patches from the input of feature attention module as de-convolutional filters to reconstruct the new feature map.

3) PUTTING IT ALL TOGETHER

Iterative inpainting is built on top of the feature extraction stage with the feature pyramid as its input. This is illustrated in the middle column of Figure 1. The pyramid is first split into two halves; feature maps coming from the first three feature modules (i.e., the first three from the input side) are concatenated to form the feature volume $\mathbf{F}_{\text{low}} \in \mathbb{R}^{C_{in} \times H_{in} \times W_{in}}$ with C_{in} channels, H_{in} height, and W_{in} width. Feature maps coming from the last three feature modules form another feature volume denoted $\mathbf{F}_{\text{high}} \in \mathbb{R}^{C_{in} \times H_{in} \times W_{in}}$. Those two volumes are sent down two different but parallel iterative branches that have the same composition of layers. Both start with two partial convolutions with leaky ReLU activations, followed by a feature attention module. The specifications of each layer are detailed at the bottom of the middle column of Figure 1.

Each branch processes the input volume iteratively, which is done as follows. Let τ represent a time index for the iterative process. Both $\mathbf{F}_{\text{low}}(0)$ or $\mathbf{F}_{\text{high}}(0)$ goes through the partial convolutions and the attention module, making up the first iteration ($\tau = 1$). The outputs, denoted $\mathbf{F}_{\text{low}}(\tau + 1)$ and $\mathbf{F}_{\text{high}}(\tau + 1)$, are used to initialize the next iteration as well as construct a new feature volume. A copy of $\mathbf{F}_{\text{low}}(\tau + 1)$ and $\mathbf{F}_{\text{high}}(\tau + 1)$ is sent back to the input to undergo the next iteration. Another copy is sent forward to a concatenation operation to form part of a new feature volume denoted $\mathbf{F}_{\text{cat}}$. This keeps on going for T iterations ($\tau \in \{1, 2, \ldots, T\}$) until $\mathbf{F}_{\text{cat}}$ is complete, i.e., a tensor of dimensions $C_{\text{cat}} \times H_{\text{cat}} \times W_{\text{cat}}$ where $C_{\text{cat}} = 2TC_{in}$ is formed. This tensor is, finally, passed to a convolution layer with leaky ReLU activation, which generates the intermediate feature volume $\mathbf{F}_{\text{int}}$.

Remark: Please note that the $\mathbf{F}_{\text{int}}$ feature volume comprises $C_{\text{int}} = C_{\text{cat}} = 2TC_{in}$ feature maps, which could be split into T sub-volumes. This is important for the sake of the third and final stage of the proposed architecture.

D. ENHANCEMENT AND RECONSTRUCTION STAGE

1) REINPAINTING COMPONENT

The main idea behind the reinpainting component is to re-enhance the fused feature sub-volumes in $\mathbf{F}_{\text{int}}$. This is done along two branches that process two different concatenations of feature sub-volumes from $\mathbf{F}_{\text{int}}$. See Figure 1. Let $\mathbf{F}_{\text{int}}(\tau)$ represent the τ-th sub-volume in $\mathbf{F}_{\text{int}}$, where $\tau \in \{1, \ldots, T - 1\}$. The first branch concatenates $\mathbf{F}_{\text{int}}(\tau - 1)$, $\mathbf{F}_{\text{int}}(\tau)$, and $\mathbf{F}_{\text{int}}(\tau + 1)$ and passes them into three convolutional layers with ReLU activations. The result is multiplied with the updated mask of iteration $\tau - 1$ from the second stage, i.e., $\tilde{\mathbf{H}}(\tau - 1)$, to eliminate the negative effect of the unfilled region in each iteration.

The second branch is symmetric with the first, but focuses on different sub-volumes. It concatenates $\mathbf{F}_{\text{int}}(\tau)$ and $\mathbf{F}_{\text{int}}(\tau + 1)$ and passes them through three convolutional layers with ReLU activations. The result here is multiplied with the difference of two updated masks from iterations τ and $\tau - 1$, i.e., $\tilde{\mathbf{H}}(\tau) - \tilde{\mathbf{H}}(\tau - 1)$, which only contains information from the intersection region between $\mathbf{F}_{\text{int}}(\tau)$ and $\mathbf{F}_{\text{int}}(\tau - 1)$. The results of the two branches are combined with the sub-volume $\mathbf{F}_{\text{int}}(\tau)$ to produce a new sub-volume $\mathbf{F}_{\text{reinp}}(\tau)$.

2) RECONSTRUCTION COMPONENT

The reinpainting model outputs a feature volume $\mathbf{F}_{\text{reinp}}$ that is fed to the reconstruction component. This is the final component of the proposed architecture, responsible for producing the complete image. A visualization of the reconstruction component is illustrated in the left panel of Fig. 1. This component adopts the feature merging module from [17] which fuses the feature group based on the filled locations in each iteration. The merge module feeds into three upsampling layers followed by a partial convolution layer, three residual blocks, and a sequence of three convolutional layers. The complete architecture is summarized in Algorithm 1.[4]

Algorithm 1 The Proposed Inpainting Network

Input Input image I_{in},
      Input structural I_{struc}, Input mask M_{in},
      Total number of iteration T

Output Recovered image I_{out}

1: $\mathbf{F}_{\text{high}}(0), \mathbf{F}_{\text{low}}(0), \tilde{\mathbf{H}}(0) \leftarrow GLE(I_{in}, I_{struc}, M_{in})$
2: $FeaturePool \leftarrow \{\mathbf{F}_{\text{high}}(0), \mathbf{F}_{\text{low}}(0)\}$
3: $\tau \leftarrow 0$
4: **if** $\tau \leq T$ **then**
5:    $\mathbf{F}_{\text{low}}(\tau + 1), \tilde{\mathbf{H}}(\tau + 1) \leftarrow \mathcal{I}(\mathbf{F}_{\text{low}}(\tau), \tilde{\mathbf{H}}(\tau))$
6:    $\mathbf{F}_{\text{high}}(\tau + 1), \tilde{\mathbf{H}}(\tau + 1) \leftarrow \mathcal{I}(\mathbf{F}_{\text{high}}(\tau), \tilde{\mathbf{H}}(\tau))$
7:    $\mathbf{F}_{\text{int}}(\tau + 1) \leftarrow FeatureFuse(\mathbf{F}_{\text{high}}(\tau + 1), \mathbf{F}_{\text{low}}(\tau + 1))$

8:    $\tau \leftarrow \tau + 1$
9: **end if**
10: $\tau \leftarrow 1$
11: **if** $\tau \leq T - 1$ **then**
12:    $F = \{\mathbf{F}_{\text{int}}(\tau - 1), \mathbf{F}_{\text{int}}(\tau), \mathbf{F}_{\text{int}}(\tau + 1)\}$
13:    $H = \{\tilde{\mathbf{H}}(\tau - 1), \tilde{\mathbf{H}}(\tau), \tilde{\mathbf{H}}(\tau + 1)\}$
14:    $\mathbf{F}_{\text{reinp}}(\tau) \leftarrow \mathcal{RI}(F, H) + \mathbf{F}_{\text{int}}(\tau)$
15:    $FeaturePool \leftarrow FeaturePool + \{\mathbf{F}_{\text{reinp}}(\tau)\}$
16:    $\tau \leftarrow \tau + 1$
17: **end if**
18: $F_{merged} \leftarrow FeatureMerge(FeaturePool)$
19: $I_{out} \leftarrow Reconstruction(F_{merged})$
20: **return** I_{out}

---

[4]To simplify the notation in Algorithm 1, We use $\mathcal{I}$ to express Iterative Inpainting Module, use $\mathcal{RI}$ to express Re-Inpainting Module.

E. LOSS FUNCTIONS

This section describes the loss functions for training the proposed inpainting network. It is a composite loss with multiple terms accounting for different aspects that the proposed algorithm needs to maintain. Perceptual loss and style loss are two of those terms that are popular for solving image generation problems. They are calculated using groundtruth and output feature maps obtained from a pretrained VGG model [25]. *Groundtruth features* are those produced by the max-pooling layers of the VGG network when the input is the complete groundtruth image whereas *output features* are those obtained from the same pooling layers but with the restored image as an input. Formally, the perceptual loss is given by

$$L_{\text{perc}} = \sum_{\theta=1}^{N} \frac{1}{H_\theta W_\theta C_\theta} |\phi_\theta^{gt} - \phi_\theta^{out}|_1, \quad (13)$$

and the style loss is given by

$$L_{\text{sty}} = \sum_{\theta=1}^{N} \frac{1}{C_\theta \times C_\theta} \left| \frac{1}{H_\theta W_\theta C_\theta} (\phi_\theta^{gt}(\phi_\theta^{gt})^T \right.$$
$$\left. - \phi_\theta^{out}(\phi_\theta^{out})^T) \right|_1, \quad (14)$$

where ϕ_θ^{gt} denotes the vectorized groundtruth feature map from the θ_{th} pooling layer of VGG-16, ϕ_θ^{out} denotes the vectorized output feature map from the θ_{th} pooling layer of pretrained VGG-16, and C_θ, H_θ, and W_θ are, respectively, the number of channels, height, and width of the θ_{th} feature map.

The third term of the composite loss is the total variation loss, which enforces smoothness in the region of predicted pixels (i.e., the holes) [24], [26]. Formally, this term is formulated as follows. Let $\mathbf{I}_{\text{out}}^{i,j}$ denote the pixel value of output image at location i, j, N denote the total number of elements in the output image, $\mathcal{R}$ denote the set of pixels surrounding a corrupted pixel $\mathbf{I}_{\text{out}}^{i,j}$. The total variation loss is given by

$$L_{tv} = \sum_{(i,j)\in R, (i,j+1)\in R} \frac{|I^{i,j+1} - I^{i,j}|_1}{N}$$
$$+ \sum_{(i,j)\in R, (i+1,j)\in R} \frac{|I^{i+1,j} - I^{i,j}|_1}{N}. \quad (15)$$

The last two terms in the composite loss are first norms of the difference between the output and groundtruth images. Let I_{out} denote the output image from the proposed algorithm, I_{gt} denote the ground truth image, and $\tilde{\mathbf{H}}_{gt}$ denote the groundtruth mask of the image. The two terms are, then, given by

$$L_{\text{valid}} = |I_{out} \odot \tilde{\mathbf{H}}_{gt} - I_{gt} \odot \tilde{\mathbf{H}}_{gt}|_1, \quad (16)$$
$$L_{\text{hole}} = |I_{out} \odot (1 - \tilde{\mathbf{H}}_{gt}) - I_{gt} \odot (1 - \tilde{\mathbf{H}}_{gt})|_1, \quad (17)$$

where L_{valid} expresses the first norm loss between undamaged region of the output image and the ground truth image, and L_{hole} expresses the first norm loss between filled region of
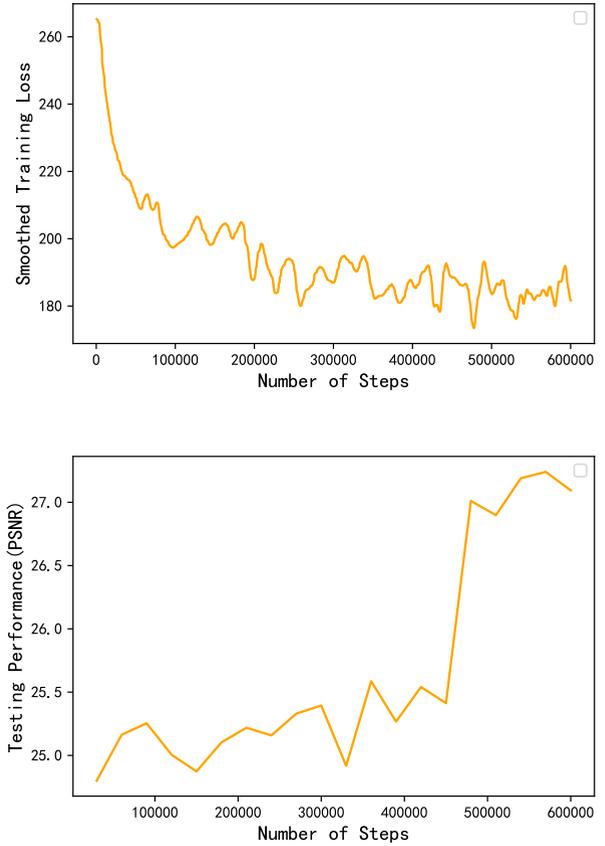
FIGURE 2. The history graph of Smoothed training loss and testing performance vs. number of steps on Celeba dataset.

the output image and the ground truth image. The composite loss, as the name suggests, is a weighted sum of all the above terms

$$\mathcal{L} = \lambda_{\text{valid}}L_{\text{valid}} + \lambda_{\text{hole}}L_{\text{hole}} + \lambda_{\text{perc}}L_{\text{perc}}$$
$$+ \lambda_{\text{style}}L_{\text{style}} + \lambda_{tv}L_{tv}, \quad (18)$$

where $\lambda_{\text{valid}}, \lambda_{\text{hole}}, \lambda_{\text{perc}}, \lambda_{\text{style}}$, and λ_{tv} are all hyperparameters scaling the contribution of each of their respective terms to the composite loss.

IV. EXPERIMENTAL SETUP

The proposed algorithm needs to be put to test in order to demonstrate its performance. This section presents the experimental setup adopted to evaluate its performance. It describes the development datasets, the implementation details, and the benchmark algorithms.

A. DATASETS

Four development datasets are adopted here:

- **Paris Streetview Dataset** [27] is collected from Google StreetView, a large-scale dataset that includes street images for 12 cities across the world. This dataset contains 15000 images, 14900 images for training, and 100 for testing.

TABLE 1. Numerical comparisons on three datasets.

		PSNR				SSIM				Mean l_1			
	Mask Ratio	10-20%	30-40%	40-50%	50-60%	10-20%	30-40%	40-50%	50-60%	10-20%	30-40%	40-50%	50-60%
PVS	PIC [31]	29.02	24.82	23.71	19.61	0.930	0.795	0.757	0.524	0.0134	0.0327	0.0429	0.0785
	PC [24]	30.42	25.35	23.86	21.47	0.933	0.825	0.763	0.628	0.1001	0.0253	0.0373	0.0612
	EC [11]	30.83	25.94	24.03	21.76	0.945	0.849	0.781	0.653	0.0084	0.0220	0.0297	0.0578
	PRVS [20]	30.87	26.21	24.11	21.94	0.952	0.857	0.794	0.665	0.0085	0.0217	0.0288	0.0567
	RFR [17]	31.74	26.35	24.56	22.43	0.954	0.864	0.811	0.687	0.0076	0.0211	0.0280	0.0543
	MDEFE [32]	31.11	26.23	24.18	22.17	0.954	0.859	0.806	0.674	0.0081	0.0217	0.0287	0.0549
	Ours	**31.99**	**26.86**	**25.04**	**22.61**	**0.964**	**0.889**	**0.833**	**0.721**	**0.0073**	**0.0192**	**0.0268**	**0.0485**
Celeba	PIC [31]	30.69	24.65	21.40	19.16	0.965	0.875	0.739	0.667	0.0094	0.0258	0.0433	0.0756
	PC [24]	32.72	26.59	24.24	22.03	0.970	0.915	0.864	0.782	0.0062	0.0181	0.0269	0.0537
	EC [11]	32.47	26.61	24.43	21.42	0.974	0.921	0.873	0.746	0.0065	0.0185	0.0260	0.0564
	PRVS [20]	33.12	27.05	24.82	22.32	0.977	0.928	0.883	0.793	0.0059	0.0169	0.0252	0.0493
	RFR [17]	33.58	27.62	25.48	22.59	0.980	0.931	0.896	0.822	0.0057	0.0161	0.0224	0.0482
	MDEFE [32]	33.20	27.21	24.97	22.44	0.976	0.925	0.879	0.814	0.0059	0.0167	0.0251	0.0509
	Ours	**33.82**	**27.93**	**25.92**	**22.76**	**0.984**	**0.945**	**0.915**	**0.834**	**0.0053**	**0.0150**	**0.0213**	**0.0359**
Place2	PIC [31]	26.67	21.24	19.13	17.27	0.931	0.761	0.656	0.499	0.0137	0.0342	0.0573	0.0953
	PC [24]	26.84	22.25	20.09	18.33	0.935	0.768	0.722	0.538	0.0131	0.0312	0.0475	0.0861
	EC [11]	27.02	22.41	20.35	18.41	0.934	0.811	0.745	0.531	0.0135	0.0309	0.0447	0.0846
	PRVS [20]	27.35	22.73	20.51	18.72	0.937	0.825	0.773	0.569	0.0140	0.0293	0.0423	0.0769
	RFR [17]	27.90	23.27	21.34	18.87	0.945	0.837	0.782	0.589	0.0114	0.0284	0.0402	0.0726
	MDEFE [32]	27.55	22.92	21.01	18.65	0.941	0.832	0.775	0.582	0.0126	0.0297	0.0411	0.0737
	Ours	**28.89**	**23.66**	**21.94**	**19.59**	**0.957**	**0.864**	**0.800**	**0.667**	**0.0099**	**0.0266**	**0.0368**	**0.0552**

[1] The best performances are highlighted in **bold**.

- **Large-scale CelebFaces Attributes (CelebA) Dataset** [28] is a well-known and publicly available face recognition dataset. It includes around 200K celebrity images representing 10000 different identities, all of which have a wide range of posture variations. This dataset contains 202599 images, 162770 images for training, 19867 images for validation, and 19962 images for testing.
- **Place2 Dataset** [28] This dataset contains 8 million images which are collected from 365 scene categories, like streets, indoor rooms and so on.
- **NVIDIA Irregular Mask Dataset Dataset** [29] is a popular irregular mask dataset. This dataset contains 12000 irregular masks which are randomly drawn by individuals. The mask ratio of the dataset is uniformly distributed on 0.0~0.1, 0.1~0.2, 0.2~0.3, 0.3~0.4, 0.4~0.5, and 0.5~0.6. Each mask ratio class has 1000 masks with and without border.

B. IMPLEMENTATION DETAILS

The proposed algorithm is trained with batch size of 4 on two NVIDIA 1080 TITANs. We use corrupted images, structural maps, and irregular holes as inputs, which are resized to 256×256. Adam [30] optimizer is used to train the network. The training is conducted with a learning rate of 10^{-4}, and the network is fine-tuned with a learning rate of 10^{-5}. The network is trained on Paris and CelebA Dataset for 40 epochs and fine-tuned for 20 epochs. On Place2 Dataset, the network is trained for 200 epochs and fine-tuned for 100 epochs. During the fine-tuning, only the weights of batch normalization layers are frozen while the rest are adjusted. The hyperparameters of the loss function are set to $\lambda_{valid} = 1$, $\lambda_{hole} = 6$, $\lambda_{perc} = 0.05$, $\lambda_{style} = 120$, and $\lambda_{tv} = 0.1$. History graphs of smoothed training loss and testing performance vs. number of steps on Celeba dataset are shown in Fig.2. The total number of parameters of the proposed network is 82 million. This means its memory footprint assuming float-32 representation is roughly 312 MB and its inference time averages 0.147 ms per image.

C. BENCHMARK ALGORITHMS AND EVALUATION METRICS

The proposed algorithm is compared to five state-of-the-art methods, namely PIC [31], PC [24], PRVS [20], EC [11], RFR [17] and MDEFE [32]. We use PIC as baseline which is a probabilistically principled framework in image inpainting. PC is a fundamental technique that can be considered as another baseline in image inpainting. EC is a two-stage image inpainting method based on the edge recovering method. PRVS and RFR belong to the family of progressively image inpainting. The PRVS progressively recover the image structural information while RFR recover the image contextual information. Those five algorithms are henceforth referred to as the benchmark algorithms. The proposed algorithm is compared to all five using three metrics, which are Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) and mean first norm loss (L_1).

V. EVALUATION RESULTS

The performance of the proposed algorithm is evaluated in this section using the setup described in Section IV. The evaluation starts with quantitative analysis where the proposed algorithm is benchmarked to others. Then, a qualitative analysis follows. It presents a comparison of the quality of inpainted images between the proposed algorithm and the benchmark algorithms. Finally, this section is concluded with an ablation analysis illustrating the value of each novel component in the proposed network.

A. QUANTITATIVE ANALYSIS

The proposed algorithm is compared to the benchmark algorithms on the basis of PSNR, SSIM, and L_1 loss. Table 1 presents the comparison results on the Paris StreetView,

| **(a)** Input | **(b)** PIC [31] | **(c)** PC [24] | **(d)** EC [11] | **(e)** PRVS [20] | **(f)** RFR [17] | **(g)** Ours | **(h)** GT |
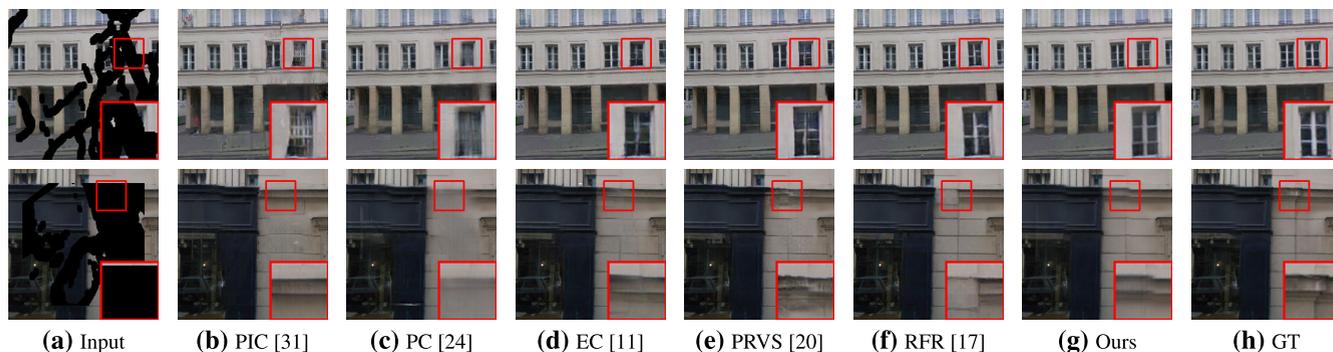
FIGURE 3. Visual results on Paris StreetView.

| **(a)** Input | **(b)** PIC [31] | **(c)** PC [24] | **(d)** EC [11] | **(e)** PRVS [20] | **(f)** RFR [17] | **(g)** Ours | **(h)** GT |
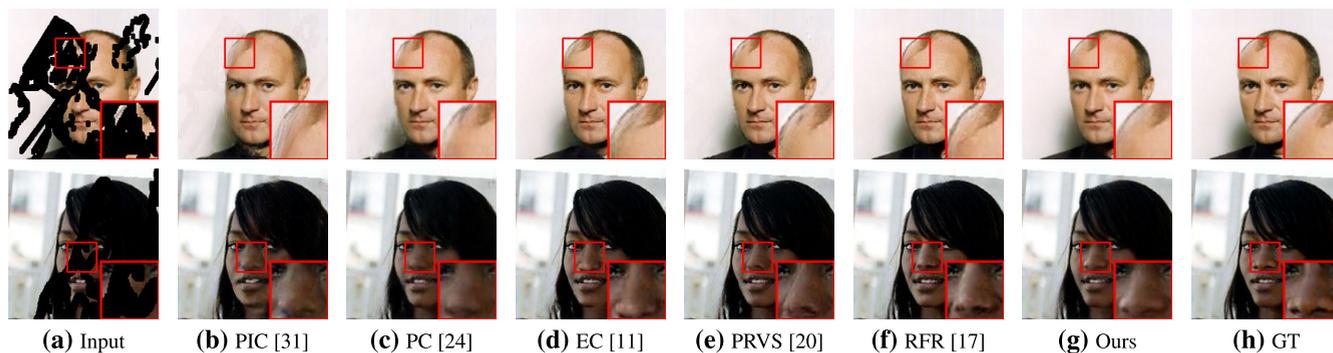
FIGURE 4. Visual results on CelebA.

| **(a)** Input | **(b)** PIC [31] | **(c)** PC [24] | **(d)** EC [11] | **(e)** PRVS [20] | **(f)** RFR [17] | **(g)** Ours | **(h)** GT |
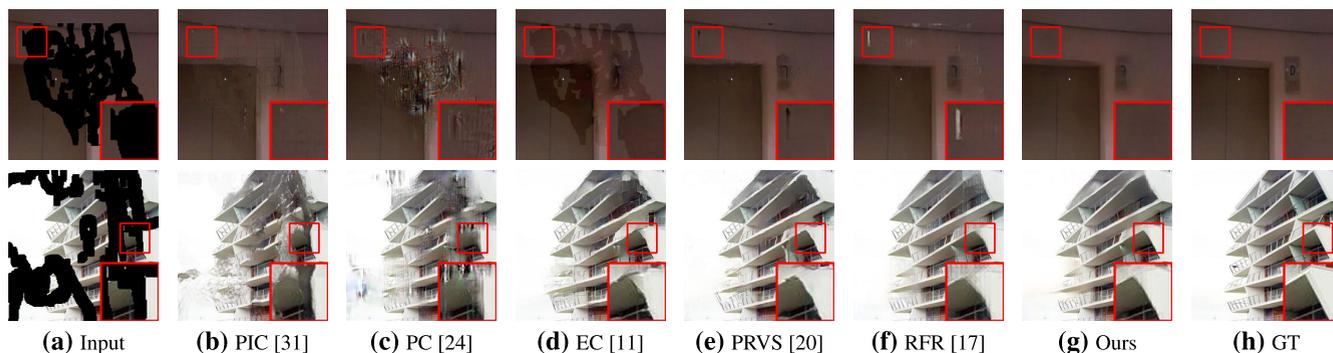
FIGURE 5. Visual results on Place2.

CelebA, and Place2 datasets. It presents the results for different choices of masking percentage (i.e., mask ratio). The performance of the proposed algorithm stands out throughout the table; despite the slim margin in some cases, its performance could be argued to best all other competing algorithms on all three datasets.

B. QUALITATIVE ANALYSIS

The above quantitative results are translated into visual analysis to demonstrate the inpainting quality of the proposed algorithm. This is done through a few examples from three datasets, namely Paris StreetView, CelebA, and Place2 datasets. Fig. 3, 4 and 5 show three corrupted images, their

groundtruth, the inpainted images by the proposed and benchmark algorithms. The proposed algorithm can generate realistic details and structures. Specifically, in the top row of Fig. 3, the window produced by the proposed algorithm is clearer than those produced by other methods. Further evidence could be seen in the top row of Fig. 4 and both rows of Fig. 5. In the former, the hair strands atop the man's forehead are better defined and clearer in the image produced by the proposed algorithm compared to those produced by the benchmark algorithms; they look similar to those strands depicted in the groundtruth image. Both rows of Fig.5 show artifacts in the inpainted region by benchmark algorithms while the proposed algorithm does not suffer

from such artifacts, producing a more pleasing image to the eye.

C. ABLATION ANALYSIS ON PROPOSED ARCHITECTURE

The proposed architecture is closely examined to get a better understanding of the role of the novel components. More to the point, the GLE module and the reinpainting component are novel parts that set the proposed architecture apart form the other inpainting algorithms. Therefore, this section will focus on shedding some light on their roles in the inpainting process. The objective is to address the question: *how much of an impact do the GLE module and the reinpainting component have on the performance of the algorithm?* This is going to be done in three experiments. The first has the two parts removed and the performance of the remaining architecture is evaluated. This helps establish the baseline results. The other two experiments examine the impact of adding each of the two parts, i.e., GLE and reinpainting, separately on the inpainting performance. The results of the five experiments are shown below.

1) REMOVING THE GLE MODULE AND REINPAINTING COMPONENT

The GLE module and reinpainting component are both removed from the proposed architecture. To avoid jeopardizing the capacity of the proposed model, the GLE module is removed by stripping away the Gaussian smoothing and upsampling layers making a direct path from the first to the second convolution layers of the module.

Removing both parts chips away from the inpainting performance of the architecture. This is evident in Table 2; with all three metrics, the table shows a clear degradation in performance on the Paris StreetView dataset when the architecture is trained and tested without the GLE module and the reinpainting component.

2) REMOVING THE GLE MODULE

Using the same removal strategy in the above section (Section V-C1), the GLE module is removed in this experiment while keeping the reinpainting component. The result of doing so is a slight improvement in the performance compared to the baseline case, i.e., no GLE and reinpainting, as Table 2 shows. However, the performance is still worse compared to having both parts plugged in. The results of this experiment could be used to argue for the value of the GLE module; it helps the proposed architecture extract expressive features from different frequency components of the image.

3) REMOVING THE REINPAINTING COMPONENT

Using the same removing strategy once again, the reinpainting component is removed while keeping the GLE module. This setting is labelled as Reinpainting-1 in Table 2. It is hypothesized that reinpainting has the ability to fill large holes by accessing features from neighboring iterations. The results in Table 2 verify that hypothesis to some extent; removing the reinpainting component degrades the

performance of the architecture despite the presence of the GLE module.

4) VALUE OF PROGRESSION FOR REINPAINTING

The hypothesis about the reinpainting component being able to fill large holes is further examined here. More to the point, it will be argued that accessing sub-volumes from different iterations (i.e., $\mathbf{F}_{\text{int}}(\tau - 1)$ and $\mathbf{F}_{\text{int}}(\tau + 1)$) has added value to the inpainting process. This is first done by restricting the input to the re-inpainting component to only the τ-th feature sub-volume, i.e., $\mathbf{F}_{\text{int}}(\tau)$. Then, two experiments are conducted with and without the GLE components.

- The GLE modules are removed as described in Section V-C1. The performance in this case is very close to that of removing the whole reinpainting component. This is indicated in Table 2 under Reinpainting-2. This verifies that the $\mathbf{F}_{\text{int}}(\tau)$ and $\tilde{\mathbf{H}}(\tau)$ can not provide more useful information for the reinpainting process. The redundant information even sightly causes the performance to degrade.

- The GLE modules are put back and the experiment is repeated again. Again, the results, shown in Table 2, further verify that the input features from neighbouring iterations are useful for enhancing the re-inpainting results.

D. ABLATION ANALYSIS ON LOSS FUNCTION

The performance of the proposed method is further investigated based on each component of the loss function. The ablation analysis is done in three steps, each of which is illustrating the incremental value of certain terms in the loss function. The three steps are discussed below:

- **Using L_{valid} and L_{hole}:** Using L_{valid} and L_{hole} as the only terms of the loss functions, we observe that the mean l_1 is slightly better compared to the final results; however, the PSNR and SSIM are not satisfactory especially for the case of corrupted images with large holes. The undesirable performance indicates that the loss has to account for the semantic information in the inpainted image.

- **Using L_{perc} and L_{style}:** Constructing the loss function using L_{perc} and L_{style} alone, we observe an improvement in the PSNR and SSIM for large holes. The performance shows the role and effectiveness of L_{perc} and L_{style} in discovering semantic information in the valid region; however, the mean l_1 loss increases significantly due to the lack of L_{valid} and L_{hole}.

- **Using L_{valid}, L_{hole}, L_{perc} and L_{style}:** After using the L_{valid}, L_{hole}, L_{perc} and L_{style} as the terms of the loss function, the PSNR, SSIM and mean l_1 all improve but for the case of low mask ratio. However, for the case of high mask ratio, the PSNR and SSIM resulting from using L_{perc} and L_{style} alone are slightly better than the case of combining all four L_{valid}, L_{hole}, L_{perc} and L_{style}. It is hypothesized that the noise may be generated from the pixel and feature levels. To eliminate such noise, the

TABLE 2. Ablation study results on the paris streetview dataset based on network structure.

Ablation setting			PSNR			SSIM			Mean l_1		
Reinpainting-1	Reinpainting-2	GLE	10-20%	30-40%	40-50%	10-20%	30-40%	40-50%	10-20%	30-40%	40-50%
✗	✗	✗	31.82	26.70	24.83	0.9632	0.8858	0.8276	0.0074	0.0196	0.0277
✗	✗	✓	31.90	26.76	24.86	0.9637	0.8859	0.8285	0.0073	0.0195	0.0275
✗	✓	✗	31.92	26.83	24.93	0.9638	0.8886	0.8314	0.0073	0.0193	0.0272
✓	✗	✗	31.93	26.83	24.92	0.9640	0.8887	0.8312	0.0073	0.0192	0.0272
✓	✗	✓	31.95	26.84	24.96	0.9641	0.8888	0.8317	0.0073	0.0192	0.0271
✗	✓	✓	31.99	26.86	25.04	0.9642	0.8890	0.8330	0.0073	0.0192	0.0268

TABLE 3. Ablation study results on the paris streetview dataset based on loss function.

Ablation setting			PSNR			SSIM			Mean l_1		
L_{valid} & L_{hole}	L_{perc} & L_{style}	L_{tv}	10-20%	30-40%	40-50%	10-20%	30-40%	40-50%	10-20%	30-40%	40-50%
✓	✗	✗	31.72	26.61	24.85	0.9604	0.8817	0.8237	0.0072	0.0193	0.0270
✗	✓	✗	31.69	26.72	24.96	0.9599	0.8830	0.8254	0.0079	0.0200	0.0278
✓	✓	✗	31.86	26.70	24.92	0.9616	0.8823	0.8223	0.0076	0.0199	0.0279
✓	✓	✓	31.99	26.86	25.04	0.9642	0.8890	0.8330	0.0073	0.0192	0.0268

L_{tv} is added to the loss function, which improves the overall performance.

VI. CONCLUSION

This paper introduces a three-stage neural network architecture that is able to progressively inpaint corrupted images while maintaining their structural and contextual integrity. In its core is a novel Gaussian-Laplacian feature Extraction (GLE) module. Stacking GLE modules constructs the first stage of the architecture and enables the network to build a feature pyramid of different frequency components, disintegrating structural (high frequency) and contextual (low frequency) information. The feature pyramid is the key for structurally- and contextually-aware progressive inpainting; low- and high-frequency components are iteratively but separately inpainted and fused in the second stage. The third, and final, stage enhances the fused features before it reconstructs the inpainted image. Experimental results and benchmarking show that the three-stage architecture is able to restore fine details in the corrupted region, outperforming the state-of-the-art algorithms. Ablation experiments reveal that the GLE module and the reinpainting component are responsible for the superior performance of the proposed architecture.

REFERENCES

[1] J. Jam, C. Kendrick, K. Walker, V. Drouard, J. G.-S. Hsu, and M. H. Yap, "A comprehensive review of past and present image inpainting methods," *Comput. Vis. Image Understand.*, vol. 203, Feb. 2021, Art. no. 103147, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314220301661

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn.*, 2000, pp. 417–424.

[3] L. Liao, R. Hu, J. Xiao, and Z. Wang, "Artist-Net: Decorating the inferred content with unified style for image inpainting," *IEEE Access*, vol. 7, pp. 36921–36933, 2019.

[4] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, and S. Satoh, "Guidance and evaluation: Semantic-aware image inpainting for mixed scenes," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 683–700.

[5] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, and S. Satoh, "Image inpainting guided by coherence priors of semantics and textures," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6539–6548.

[6] L. Liao, J. Xiao, Z. Wang, C.-W. Lin, and S. Satoh, "Uncertainty-aware semantic guidance and estimation for image inpainting," *IEEE J. Sel. Topics Signal Process.*, vol. 15, no. 2, pp. 310–323, Feb. 2021.

[7] Z. Qin, Q. Zeng, Y. Zong, and F. Xu, "Image inpainting based on deep learning: A review," *Displays*, vol. 69, Sep. 2021, Art. no. 102028.

[8] C. Guillemot and O. Le Meur, "Image inpainting: Overview and recent advances," *IEEE Signal Process. Mag.*, vol. 31, no. 1, pp. 127–144, Jan. 2014.

[9] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, p. 24, Jul. 2009.

[10] Y. Ren, X. Yu, R. Zhang, T. H. Li, S. Liu, and G. Li, "StructureFlow: Image inpainting via structure-aware appearance flow," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 181–190.

[11] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi, "Edge-Connect: Structure guided image inpainting using edge prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1–10.

[12] Z. Guo, Z. Chen, T. Yu, J. Chen, and S. Liu, "Progressive image inpainting with full-resolution residual network," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 2496–2504.

[13] Y. Zeng, Z. Lin, J. Yang, J. Zhang, E. Shechtman, and H. Lu, "High-resolution image inpainting with iterative confidence feedback and guided upsampling," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 1–17.

[14] X. Guo, H. Yang, and D. Huang, "Image inpainting via conditional texture and structure dual generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 14134–14143.

[15] C. Hsu, F. Chen, and G. Wang, "High-resolution image inpainting through multiple deep networks," in *Proc. Int. Conf. Vis., Image Signal Process. (ICVISP)*, Sep. 2017, pp. 76–81.

[16] H. Zhang, Z. Hu, C. Luo, W. Zuo, and M. Wang, "Semantic image inpainting with progressive generative networks," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 1939–1947.

[17] J. Li, N. Wang, L. Zhang, B. Du, and D. Tao, "Recurrent feature reasoning for image inpainting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7760–7768.

[18] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes, and J. Luo, "Foreground-aware image inpainting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5840–5848.

[19] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, "EdgeConnect: Generative image inpainting with adversarial edge learning," 2019, *arXiv:1901.00212*.

[20] J. Li, F. He, L. Zhang, B. Du, and D. Tao, "Progressive reconstruction of visual structure for image inpainting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5962–5971.

[21] M. S. Farid, H. Khan, and A. Mahmood, "Image inpainting based on pyramids," in *Proc. IEEE 10th Int. Conf. SIGNAL Process. Proc.*, Oct. 2010, pp. 711–715.

[22] J. H. Lee, I. Choi, and M. H. Kim, "Laplacian patch-based image synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2727–2735.

[23] S. Padmavathi and K. Soman, "Laplacian pyramid based hierarchical image inpainting," *Adv. Image Video Process.*, vol. 2, no. 1, pp. 9–22, 2014.

[24] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 85–100.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[26] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5188–5196.

[27] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros, "What makes Paris look like Paris?" *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–10, 2012.

[28] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3730–3738.

[29] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. (2018). *Nvidia Irregular Mask Dataset*. [Online]. Available: https://nv-adlr.github.io/publication/partialconv-inpainting

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[31] C. Zheng, T.-J. Cham, and J. Cai, "Pluralistic image completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1438–1447.

[32] H. Liu, B. Jiang, Y. Song, W. Huang, and C. Yang, "Rethinking image inpainting via a mutual encoder–decoder with feature equalizations," in *Proc. 16th Eur. Conf. Comput. Vis.* Glasgow, U.K.: Springer, Aug. 2020, pp. 725–741.

MUHAMMAD ALRABEIAH (Member, IEEE) received the bachelor's degree in electrical engineering from King Saud University (KSU), Riyadh, Saudi Arabia, in 2010, the master's degree in video and image processing from McMaster University, Canada, in 2015, and the Ph.D. degree from the School of ECEE, Arizona State University (ASU), Tempe, AZ, USA. He is currently a Machine Learning and Wireless Technologies Researcher, an Assistant Professor, and a Consultant. He is also a full-time AI and Wireless Technologies Consultant with the Research and Innovation Center (RIC), Communications and Information Technology Commission (CITC), Saudi Arabia. He holds a faculty position with the Electrical Engineering Department, KSU, where he is also a member of the Technology Innovation Center in RF and Photonics (RFTONICS). He has specialized in machine learning and its applications to computer vision and wireless technologies (i.e., 5/6G and wireless sensing) at ASU. His research and professional interests include the field of deep learning and its applications to computer vision, 5G/6G wireless systems, vision-aided wireless communications, and radio perception.
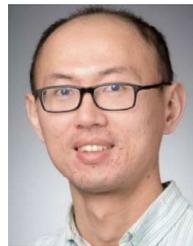
JUN CHEN (Senior Member, IEEE) received the B.S. degree in electronic engineering from Shanghai Jiao Tong University, in 2001, and the M.S. and Ph.D. degrees in electrical and computer engineering from Cornell University, in 2004 and 2006, respectively.

He was a Postdoctoral Research Associate with the Coordinated Science Laboratory, University of Illinois at Urbana–Champaign, from September 2005 to July 2006, and a Postdoctoral Fellow with the IBM Thomas J. Watson Research Center, from July 2006 to August 2007. Since September 2007, he has been with the Department of Electrical and Computer Engineering, McMaster University, where he is currently a Professor.

Dr. Chen held the title of the Barber-Gennum Chair in information technology, from 2008 to 2013, and the Joseph Ip Distinguished Engineering Fellow, from 2016 to 2018. He was a recipient of the Josef Raviv Memorial Postdoctoral Fellowship, in 2006, the Early Researcher Award from the Province of Ontario, in 2010, the IBM Faculty Award, in 2010, the JSPS Invitational Fellowship, in 2020, and the ICC Best Paper Award, in 2020. He served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY, from 2014 to 2016. He is also an Editor of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING.

• • •

KANGDI SHI received the B.E. degree in electrical and computer engineering from the University of Manitoba, Winnipeg, Canada, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, Canada. His research interests include image classification, video super-resolution, video coding, and image inpainting.