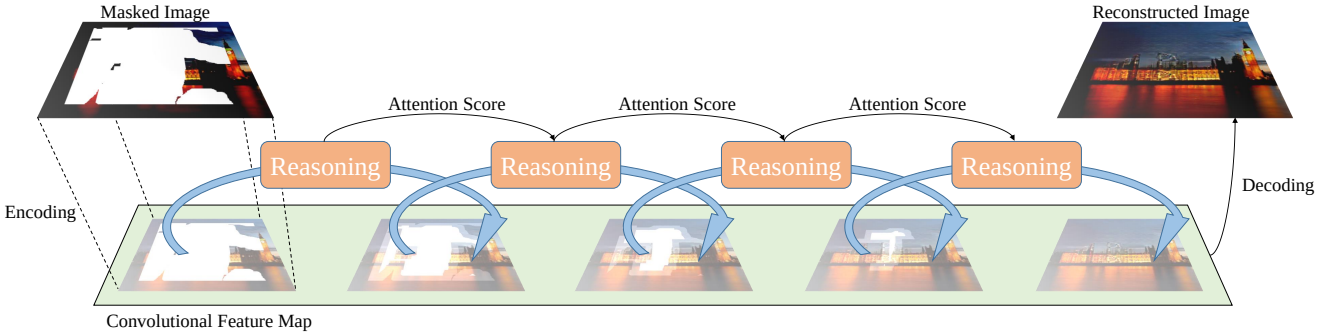
Recurrent Feature Reasoning for Image Inpainting

Jingyuan Li[1], Ning Wang[1], Lefei Zhang*[1], Bo Du[1], Dacheng Tao[2]

[1] National Engineering Research Center for Multimedia Software, School of Computer Science and Institute of Artificial Intelligence, Wuhan University, China
[2] UBTECH Sydney AI Centre, School of Computer Science, Faculty of Engineering, The University of Sydney, Darlington, NSW 2008, Australia
{jingyuanli, wang_ning, zhanglefei, dubo}@whu.edu.cn, dacheng.tao@sydney.edu.au



The overview of our proposed inpainting scheme. The masked image is first mapped into the convolutional feature space and processed by a shared Feature Reasoning module recurrently. After the feature map is fully recovered, the generated feature maps are merged together (Omitted in this figure) and the merged feature is translated back to a RGB image.

最后进行融合，

Abstract

Existing inpainting methods have achieved promising performance for recovering regular or small image defects. However, filling in large continuous holes remains difficult due to the lack of constraints for the hole center. In this paper, we devise a ***Recurrent Feature Reasoning*** *(RFR) network which is mainly constructed by a plug-and-play Recurrent Feature Reasoning module and a* ***Knowledge Consistent Attention*** *(KCA) module. Analogous to how humans solve puzzles (i.e., first solve the easier parts and then use the results as additional information to solve difficult parts), the RFR module recurrently infers the hole boundaries of the convolutional feature maps and then uses them as clues for further inference. The module progressively strengthens the constraints for the hole center and the results become explicit. To capture information from distant places in the feature map for RFR, we further develop KCA and incorporate it in RFR. Empirically, we first compare the proposed RFR-Net with existing backbones, demonstrating that RFR-Net is more efficient (e.g., a 4% SSIM improvement for the same model size). We then place the network in the context of the current state-of-the-art, where it exhibits improved performance. The corresponding source code is available at:* https://github.com/jingyuanli001/RFR-Inpainting

1. Introduction

Image inpainting aims to recover the missing regions of damaged images with realistic contents. These algorithms have a wide range of applications in photo editing, de-captioning and other scenarios where people might want to remove unwanted objects from their photos [23, 22, 1, 4]. A successfully inpainted image should exhibit coherence in both structure and texture between the estimated pixels and the background [27, 17].

Recently, deep convolutional networks have been used to solve the inpainting problem. Most state-of-the-art (SOTA) methods [29, 17, 25, 30] exploit an encoder-decoder architecture and assume that the damaged image, once encoded,

*Corresponding Author

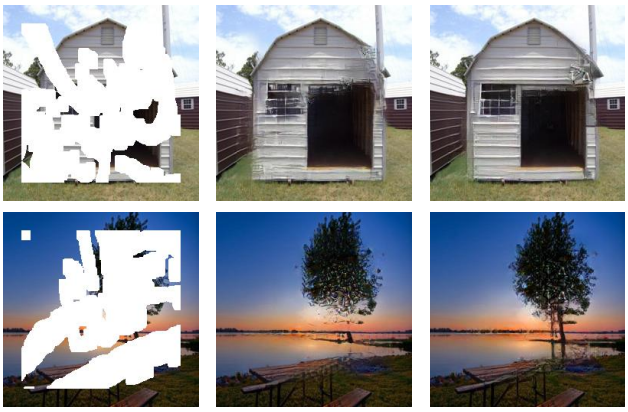


Figure 1. Illustration of semantic ambiguity that existing one-shot inpainting methods suffer from. Due to the lack of explicit constraints on the hole center, it is challenging for a network to directly inpaint on the hole center. The results of One-Shot Fill are taken from the state-of-the-art method [11].

should have adequate information for reconstruction and then inpaint in one shot. This assumption is reasonable for small or narrow defects, because pixels within local areas can have strong correlations and a pixel can therefore be inferred from its surroundings. However, as the damaged areas become larger and the distances between known and unknown pixels increase, these correlations are weakened and the constraints for the hole center loosen. Under such circumstances, the information in the known area is not informative for recovering the pixels at the center of the hole and the networks generate semantically ambiguous results (see Fig 1). An alternative scheme is to inpaint in a progressive fashion from the hole boundary to the center [31, 9]. However, these methods do not use recurrent designs and render redundant models. Also, because progressive inpainting is performed at the image level, the computational cost makes these methods less practical. Further, their inpainting schemes are only feasible in one-stage methods but unsuitable for multi-stage methods whose sub-networks cannot meet the design requirement that inputs and outputs are represented in the same space (e.g., RGB space). Finally, the process of mapping the feature map back to the RGB space occurs in each iteration, which results in information distortion in every recurrence (e.g., transforming a $128\times128\times64$ feature map into a $256\times256\times3$ RGB image). As a result, they either underperform or have an unacceptably high computational cost.

In this paper, we propose a new deep image inpainting architecture called the Recurrent Feature Reasoning Network (RFR-Net). Specifically, we devise a plug-and-play Recurrent Feature Reasoning (RFR) module to recurrently infer and gather the hole boundary for the encoded feature map. In this way, the constraints determining the internal contents are progressively strengthened and the model can produce semantically explicit results. Unlike existing progressive methods, the RFR module performs this progressive process in the feature map space, which not only ensures superior performance but also addresses the limitation that the inputs and outputs of the network need to be represented in the same space. The recurrent design reuses the parameters to deliver a much lighter model. Also, the computational cost can be flexibly controlled by moving the module up and down in the network. This is essential for building high-resolution inpainting networks, because avoiding the computation of the first and last few layers can eliminate most of the computational burden.

To further reinforce RFR's potential for recovering high-quality feature maps, attention modules [29] are necessary. However, directly applying existing attention designs in the RFR is suboptimal, because they fail to consider the consistency requirements between feature maps in different recurrences. This could lead to blurred texture in recovered area. To overcome this problem, we devise a Knowledge Consistent Attention (KCA) mechanism, which shares the attention scores between recurrences and adaptively combines them to guide a patch-swap process. Assisted by the KCA, the level of consistency is enhanced and the model's performance improves.

Our main contributions are as follows:

- We propose a Recurrent Feature Reasoning (RFR) module, which exploits the correlation between adjacent pixels and strengthens the constraints for estimating deeper pixels. The RFR module not only significantly enhances network performance but also bypasses several limitations of progressive methods.
- We develop a Knowledge Consistent Attention (KCA) module which adaptively combines the scores from different recurrences and ensures consistency between patch-swapping processes among recurrences, leading to better results with exquisite details.
- The new modules are assembled and form a novel RFR network. We analyze our model in terms of both efficiency and performance and demonstrate RFR's superiority to several state-of-the-art methods in benchmark datasets.

2. Related Works

In this section, we summarize some previous work related to our method.

Image Inpainting Traditionally, inpainting algorithms attempt to find patches from the background area to inpaint the hole [1, 2, 3, 5, 12, 23, 7]. Although these traditional methods are very good for simple cases, they cannot inpaint on complex scenes due to a lack of semantic understanding of the image.

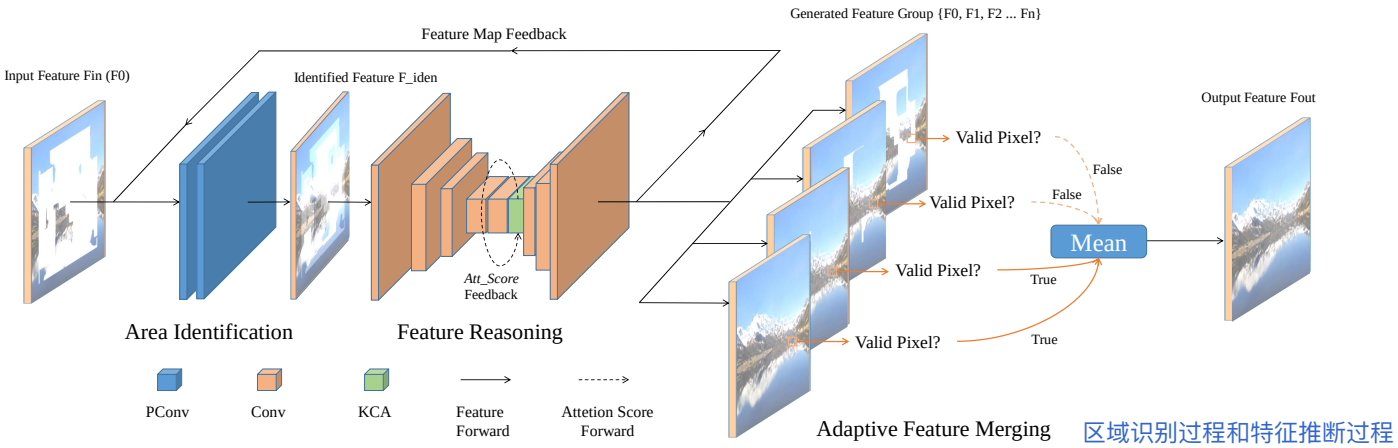


Figure 2. Illustration of the Recurrent Feature Reasoning module. The area identification process and the feature reasoning process are performed continuously. After several times of reasoning, the feature maps are merged in an adaptive fashion and an output feature map of a fixed channel numbers are generated. The module is Plug-In-and-Play and can be placed in any layer of an existing network.

即插即用

To semantically inpaint damaged images, deep convolutional networks[28], especially generative adversarial networks (GANs) [8], have recently been used. Context-Encoder [19] first employed a conditional GAN [16] for image inpainting and demonstrated the potential of CNNs for inpainting tasks. Iizuka *et al*. [10] introduced an extra discriminator to ensure local image coherency and used Poisson blending [20] to refine the image, which rendered more detailed and sharper results. Yan *et al*. [27] and Yu *et al*. [29] devised feature shift and contextual attention operations, respectively, to allow the model to borrow feature patches from distant areas of the image. Liu *et al*. [13] and Yu *et al*. [30] devised special convolutional layers to enable the network to inpaint on irregularly masked images. These methods fail to address the semantic ambiguity as they try to recover the whole target with inadequate constraints.

Progressive Inpainting Progressive image inpainting has recently been investigated. Xiong *et al*. [26] and Nazeri *et al*. [17] filled images with contour/edge completion and image completion in a step-wise manner to ensure structural consistency. Li *et al*. [11] added progressively reconstructed boundary maps as extra training targets to assist the inpainting process of a U-Net. These approaches attempted to solve inpainting tasks by adding structural constraints, but they still suffer from the lack of information for restoring deeper pixels in holes for their backbone. Zhang *et al*. [31] used cascaded generators to progressively fill in the image. Guo *et al*. [9] directly inpainted on the original size images with a single stage feed-forward network. Oh *et al*. [18] used an onion-peel scheme to progressively inpaint on video data using content from reference frames, allowing accurate content borrowing. However, these methods generally suffered from the limitations of progressive inpainting described in the introduction.

Attentive Inpainting Image inpainting models can adopt an attention mechanism to borrow features from the background. Yu *et al*. [29] exploited textural similarities within the same image to fill defects with more realistic textures from the background area. Wang *et al*. [24] devised a multi-scale attention module to enhance the precision of the patch-swapping process. Liu *et al*. [14] used a coherent semantic attention layer to ensure semantic relevance between swapped features. Xie *et al*. [25] devised a bidirectional attention map estimating module for feature re-normalization and mask-updating during feature generation. Although these methods have delivered considerable improvements, they remain suboptimal for a recurrent architecture as they do not take relationships between the feature maps from different recurrences into consideration.

3. Method

In this section, we first introduce the RFR module, which forms the main body of RFR-Net. Then we introduce the KCA scheme, which exploit the recurrent network design. Finally, the overall architecture and corresponding target functions are introduced.

3.1. Recurrent Feature Reasoning Module

The RFR module is a plug-and-play module with a recurrent inference design that can be installed in any part of an existing network. The RFR module can be decomposed into three parts: 1) an area identification module, which is used to identify the area to be inferred in this recurrence; 2) a feature reasoning module, which aims to infer the content in the identified area; and 3) a feature merging operator, which merges the intermediate feature maps. Inside the module, the area identification module and the feature reasoning module work alternately and recurrently. After the holes are filled, all the feature maps generated during inference are merged to produce a feature map with fixed channel numbers. We elaborate these processes below. The

区域识别模块、特征推断模块、特征融合操作

model pipeline of our module is shown in Fig. 2.

3.1.1 Area Identification

Partial convolution [13] is a basic module used to identify the area to be updated in each recurrence. The partial convolutional layer updates the mask and re-normalizes the feature map after the convolution calculation. More formally, the partial convolution layer can be described as follows. Let F^* denote the feature map generated by the partial convolution layer. $f^*_{x,y,z}$ denotes the feature value at location x, y in the z^{th} channel. W_z is the z^{th} convolution kernel in the layer. $f_{x,y}$ and $m_{x,y}$ are the input feature patch and input mask patch (whose size is the same as the convolutional kernel) centered at location x, y, respectively. Then, the feature map computed by the partial convolution layer can be represented by:

$$f^*_{x,y,z} = \begin{cases} W_z^T(f_{x,y} \odot m_{x,y} \frac{sum(1)}{sum(m_{x,y})}) + b, & \text{if sum}(m_{x,y}) \text{ != 0} \\ 0, & \text{else} \end{cases} \quad (1)$$

Pconv版本

Similarly, the new mask value at location i, j generated by the layer can be expressed as:

可以同时生成置信度

$$m^*_{x,y} = \begin{cases} 1, & \text{if sum}(m_{x,y}) \text{ != 0} \\ 0, & \text{else} \end{cases} \quad (2)$$

Given the equations above, we are able to receive new masks whose holes are smaller after each partial convolutional layer.

For area identification in the RFR module, we cascade several partial convolutional layers together to update the mask and feature map. After passing though the partial convolutional layers, the feature maps are processed by a normalization layer and an activation function before being sent to the feature reasoning modules. We define the difference between the updated masks and the input masks as the areas to be inferred in this recurrence. The holes in the updated mask are preserved throughout this recurrence until being further shrunk in the next recurrence.

3.1.2 Feature Reasoning

With the area to be processed identified, the feature values in the area are estimated by the feature reasoning module. The goal of the feature reasoning module is to fill in the identified area with as high-quality feature values as possible. The high-quality features not only produce a better final result but also benefit following inferences. As a result, the feature reasoning module can be designed to be very complicated to maximize its inference capability. However, here we simply stack a few encoding and decoding layers and bridge them using skip connections so that we can intuitively show the efficiency of the feature reasoning module.

加入跳连接

After the feature values are inferred, the feature maps will be sent to the next recurrence. Since the RFR module does not constrain the representation of the intermediate results, the updated mask and the partially inferred feature maps are directly sent to the next recurrence without further processing.

3.1.3 Feature Merging

When the feature maps are completely filled (or after a specific number of recurrences), the feature maps have passed through the feature reasoning module several times. If we directly use the last feature map to generate the output, gradient vanishing can occur, and signals generated in earlier iteration are damaged. To solve this problem, we must merge the intermediate feature maps. However, using convolutional operations to do so limits the number of recurrences, because the number of channels in the concatenation is fixed. Directly summing all feature maps removes image details, because the hole regions in different feature maps are inconsistent and prominent signals are smoothed. As a result, we use an adaptive merging scheme to address the issue. The values in the output feature map are only calculated from the feature maps whose corresponding locations have been filled. Formally, let's define F^i as the i^{th} feature map generated by the feature reasoning module and $f_{x,y,z}$ as the value at location x,y,z in feature map F. M^i is the binary mask for feature map F^i. The value at the output feature map $\bar{F}$ can be defined as:

cat需要固定个数通道，不适用

加权平均一下

$$\bar{f}_{x,y,z} = \frac{\sum_{i=1}^{N} f^i_{x,y,z}}{\sum_{i=1}^{N} m^i_{x,y,z}} \quad (3)$$

where N is the number of feature maps. In this way, feature maps of arbitrary number can be merged, which gives the RFR the potential to fill larger holes.

3.2. Knowledge Consistent Attention

In image inpainting, the attention module is used to synthesize features of better quality [29]. The attention modules search for possible texture in the background and use them to replace the textures in the holes. However, directly inserting the existing attention modules into the RFR is suboptimal, because the patch swapping processes in different recurrences are performed independently. The discrepancy between the components of the synthesized feature maps can damage the feature map when they are merged (see section 3.1.3). To address the issue, we devise a novel attention module, Knowledge Consistent Attention (KCA). Unlike previous attention mechanism whose attention scores are calculated independently, the scores in our KCA are composed of scores proportionally accumulated from previous recurrences. As a result, the inconsistencies between

差异

KCA由得分累计得到

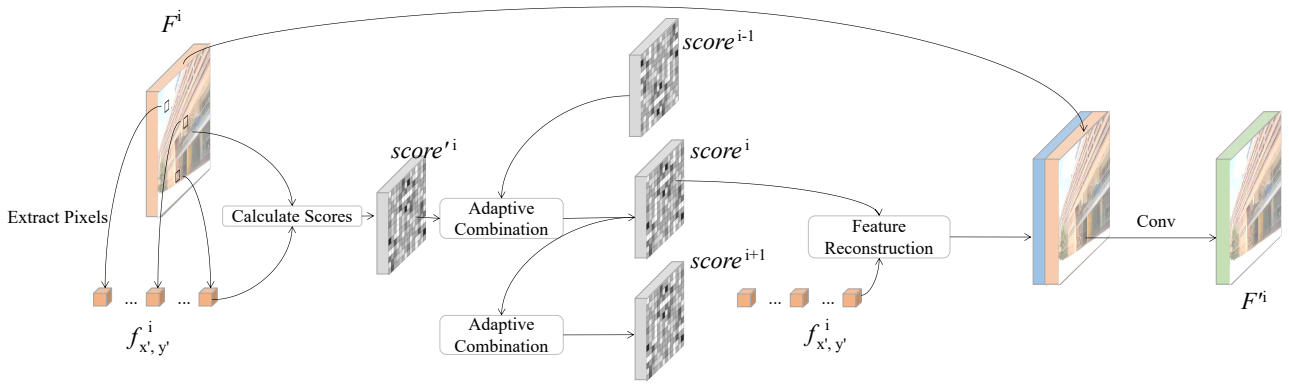


Figure 3. The illustration of the Knowledge Consistent Attention. The attention scores of the KCA are adaptively computed from the previous and current ones to ensure consistent feature ingredient.

the attention feature maps can be controlled. The illustration of the attention mechanism is given in Fig. 3, and the details are provided below.

In our design, the final components for each pixel are decided as follows. Let's denote the input feature map in the i^{th} recurrence F^i. First, we measure the cosine similarity between each pair of feature pixels:

余弦相似度

$$\widehat{sim}^{i}_{x,y,x',y'} = \langle \frac{f_{x,y}}{||f_{x,y}||}, \frac{f_{x',y'}}{||f_{x',y'}||} \rangle \tag{4}$$

where $sim^i_{x,y,x',y'}$ indicates the similarity between the hole feature at location (x, y) and that at location (x', y'). After that, we smooth the attention scores by averaging the similarity of a target pixel in an adjacent area.

局部平滑性或者一致性

$$sim'^{i}_{x,y,x',y'} = \frac{\sum_{p,q\in\{-k,\dots,k\}} \widehat{sim}_{x+p,y+q,x',y'}}{k \times k} \tag{5}$$

Then, we generate the proportion of the component for the pixel at location (x, y) using the softmax function. The generated score map is denoted $score'$.

To calculate the final attention score of a pixel, we first make a decision about whether the score of the pixel in previous recurrence will be referred to. Given a pixel considered to be valid, its attention score in current recurrence is then calculated as the weighted sum of the original and final scores from current and previous recurrences, respectively. Formally, if the pixel at location (x, y) is a valid pixel in the last recurrence (i.e., the mask value $m^{i-1}_{x,y}$ is 1), we adaptively combine the final score of the pixel from last recurrence with the score calculated in this recurrence as follows where λ is a learnable parameter:

$$score^{i}_{x,y,x',y'} = \lambda score'^{i}_{x,y,x',y'} + (1-\lambda) score^{i-1}_{x,y,x',y'} \tag{6}$$

Otherwise, if the pixel is not valid in the last recurrence, no extra operation will be performed, and the final attention score of the pixel in current recurrence will be calculated as follow:

$$score^{i}_{x,y,x',y'} = score'^{i}_{x,y,x',y'} \tag{7}$$

当前得分由之前的和现在的加权平均得到,lambda可学习

In the end, the attention score is used to rebuild the feature map. Specifically, the new feature map at location (x, y) is calculated as follows:

$$\widehat{f}^{i}_{x,y} = \sum_{x'\in 1,\dots W\, y'\in 1,\dots H} score^{i}_{x,y,x',y'} f^{i}_{x',y'} \tag{8}$$

After the feature map is reconstructed, the input feature F and the reconstructed feature map $\widehat{F}$ are concatenated and sent to a convolution layer:

$$F'^{i} = \phi(|\widehat{F}, F|) \tag{9}$$

where F' is the reconstructed feature map and ϕ is the pixel-wise convolution.

3.3. Model Architecture & Loss Functions

In our implementation, we put 2 and 4 convolution layers before and after the RFR module, respectively. We man-

Algorithm 1 The RFR Inpainting Network

Input:
Img_{in}: Input image
$Mask_{in}$: Input mask
Output:
Img_{rec}: Reconstructed image
Begin Algorithm

$F^0, M^0 \leftarrow Encoding(Img_{in}, Mask_{in})$
$FeatureGroup \leftarrow \{F^0\}$
$i \leftarrow 0$
while i smaller than $IterNum$ **do**
  $F^{i+1}, M^{i+1} \leftarrow AreaIden(F^i, M^i)$
  $F^{i+1} \leftarrow FeatReason(F^{i+1})$
  $FeatureGroup \leftarrow FeatureGroup + \{F^{i+1}\}$
  $i \leftarrow i+1$
end while
$F_{merged} \leftarrow FeatMerge(FeatureGroup)$
$Img_{rec} \leftarrow Decoding(F_{merged})$
return Img_{rec}

ually choose the recurrence number $IterNum$ to be 6 to simplify training. The KCA module is placed after the third last layer in the RFR module's feature reasoning module. The pipeline of the network is described in Alg. 1.

For image generation learning, the perceptual loss and style loss from a pre-trained and fixed VGG-16 are used. The perceptual loss and style loss compare the difference between the deep feature map of the generated image and the ground truth. Such a loss function can effectively teach the model the image's structural and textural information. These loss functions are formalized as follows. ϕ_{pool_i} denotes feature maps from the i^{th} pooling layer in the fixed VGG-16. In the following equations, H_i, W_i and C_i are used to express the height, weight and channel size of the i^{th} feature map, respectively. The perceptual loss can then be written as follows:

$$L_{perceptual} = \sum_{i=1}^{N} \frac{1}{H_i W_i C_i} |\phi_{pool_i}^{gt} - \phi_{pool_i}^{pred}|_1 \tag{10}$$

Similarly, the computation of the style loss is as follows:

$$\phi_{pool_i}^{style} = \phi_{pool_i} \phi_{pool_i}^{T} \tag{11}$$

$$L_{style} = \sum_{i=1}^{N} \frac{1}{C_i \times C_i} \Big| \frac{1}{H_i W_i C_i} (\phi_{pool_i}^{style_{gt}} - \phi_{pool_i}^{style_{pred}}) \Big|_1 \tag{12}$$

Further, L_{valid} and L_{hole} which calculate L1 differences in the unmasked area and masked area respectively are also used in our model. In summary, our total loss function is:

$$\begin{aligned} L_{total} = {} & \lambda_{hole} L_{hole} + \lambda_{valid} L_{valid} + \lambda_{style} L_{style} \\ & + \lambda_{perceptual} L_{perceptual} \end{aligned} \tag{13}$$

The loss function combination in our model is similar to [13] and has been shown to be effective in previous works [11]. This kind of loss function combination also enables efficient training due to the smaller number of parameters to update.

4. Experiments

In this section, we provide the detailed experimental settings to assist with reproducibility.

4.1. Training Setting

We train our model with batch size 6 using the Adam optimizer. Since we only have a generator network to update, no optimizer is required for the discriminator. At the start, we use a learning rate of $1e^{-4}$ to train the model and then use $1e^{-5}$ for fine-tuning the model. During fine-tuning, we freeze all the batch normalization layers. For the hyper-parameters, we use 6 for λ_{hole}, 1 for λ_{valid}, 0.1 for $\lambda_{perceptual}$, and 180 for λ_{style}. All experiments are conducted using Python on an Ubuntu 17.10 system, with an i7-6800K 3.40GHz CPU and an 11G NVIDIA RTX2080Ti GPU.

4.2. Datasets

We use three public image datasets commonly used for image inpainting tasks and a mask dataset [13] to validate our model.

Places2 Challenge Dataset [33]: A dataset released by MIT containing over 8,000,000 images from over 365 scenes, which is very suitable for building inpainting models as it enables the model to learn the distribution from many natural scenes.

CelebA Dataset [15]: A dataset focusing on human face images and containing over 180,000 training images. The model trained on this dataset can easily be transferred to face editing/completion tasks.

Paris StreetView Dataset [6]: A dataset containing 14,900 training images and 100 test images collected from street views of Paris. This dataset mainly focuses on the buildings in the city.

4.3. Comparison Models

We compare our approach with several state-of-the-art methods. These models are trained until convergence with the same experiment settings as ours. These models are: PIC [32], PConv [13], GatedConv [30], EdgeConnect [17], and PRVS [11].

5. Results

We conduct experiments on the three datasets and measure qualitative and quantitative results to compare our model with previous methods. We then compare the bare RFR-Net (without the attention module) with the existing backbone networks with respect to model size and quantitative performance to demonstrate the efficiency of our proposed method. Further, we conduct ablation studies to examine the design detail of our model.

5.1. Comparisons with State-of-the-art Methods

In this part, we compared our RFR-Net with several state-of-the-art methods mentioned in the last section. We conducted qualitative analysis and quantitative analysis respectively to demonstrate the superiority of our method.

Qualitative Comparisons Figs. 4, 5, and 6 compare our method with five state-of-the-art approaches on the Places2, CelebA, and Paris StreetView datasets, respectively. Our inpainting results have significantly fewer noticeable inconsistencies than the state-of-the-art methods in most cases, especially for large holes. Compared to the other methods, our proposed algorithm generates more semantically plausible and elegant results.

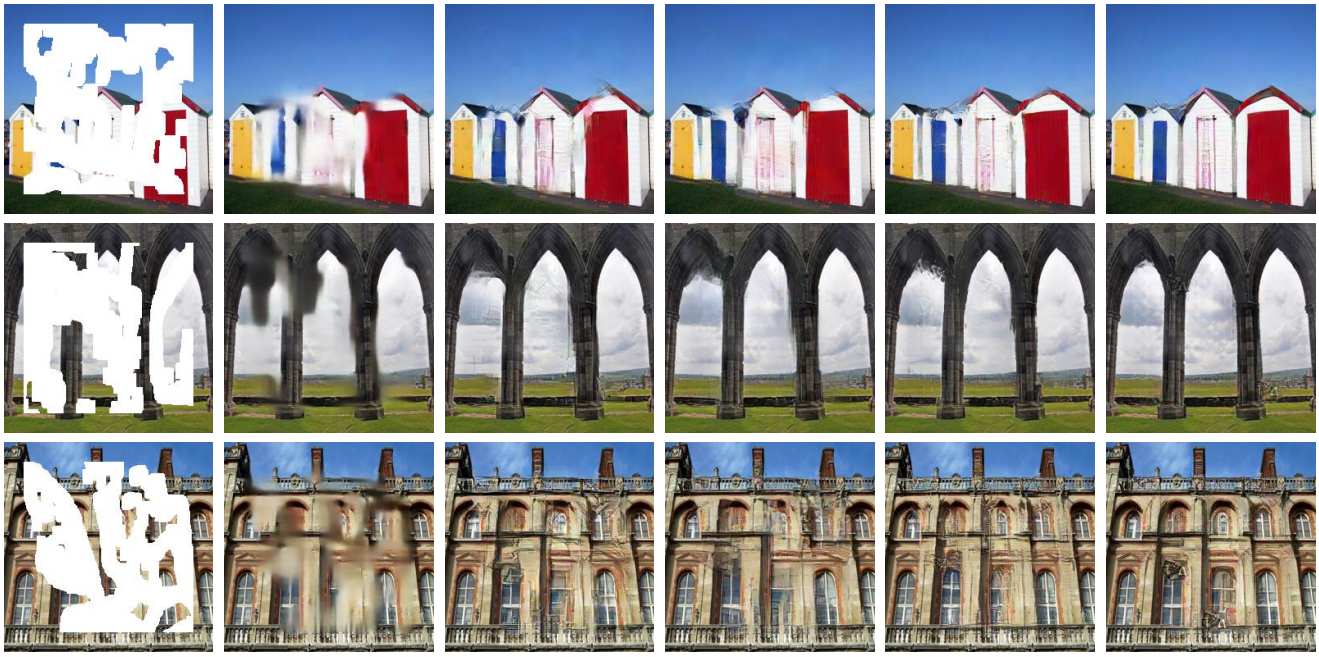


Figure 4. Results on Places2

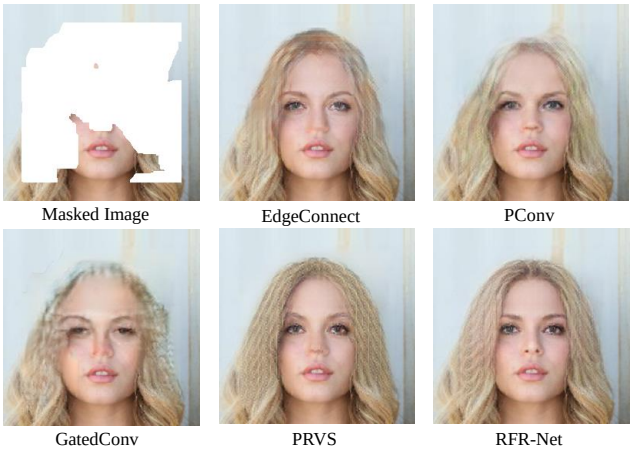


Figure 5. Results on CelebA

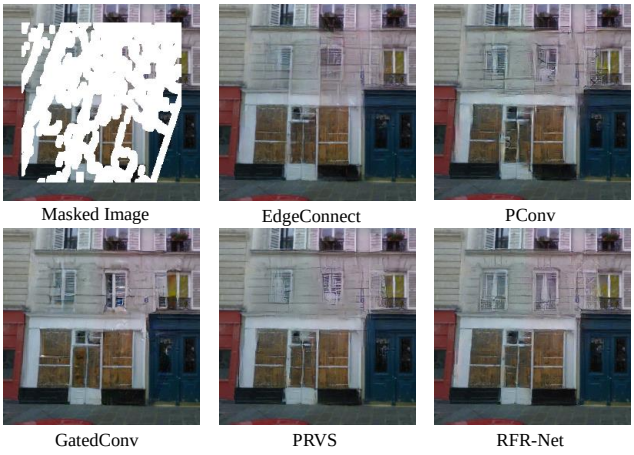


Figure 6. Results on Paris StreetView

Quantitative Comparisons We also compare our model quantitatively in terms of structural similarity index (SSIM), peak signal-to-noise ratio (PSNR) and mean l_1 loss. Table 1 lists the results with different ratios of irregular masks for the three datasets. As shown in Table 1, our method produces excellent results and the highest SSIM, PSNR and mean l_1 loss on the Places2, CelebA, and Paris StreetView datasets. The missing results in the table are due to the limitation in computational resources.

5.2. Model Efficiency

As shown in Table 2, our bare RFR-Net (without the attention module) has fewer parameters than the widely used Coarse-To-Fine [14] and PConv-UNet [13, 21] backbones. We compare the quantitative performance of these three networks (Table 2) with mask ratio of 40%-50% on Paris StreetView. For Coarse-To-Fine, we use the data reported in the CSA paper [14] because the official code is unavailable. Our bare RFR-Net produces good results, with the best SSIM and PSNR of the tested methods. The inference time of our model for each image is usually between 85 and 95 ms, which is also faster than several state-of-the-art methods (e.g. [30, 14, 11]). Taken together, we can conclude that our method can achieve better results than methods with similarly sized models, highlighting the efficiency of our proposed method.

Dataset		Places2			CelebA			Paris Street View		
Mask Ratio		10%-20%	30%-40%	50%-60%	10%-20%	30%-40%	50%-60%	10%-20%	30%-40%	50%-60%
SSIM*	PIC	0.932	0.786	0.494	0.965	0.881	0.672	0.930	0.785	0.519
	PConv	0.934	0.803	0.555	0.977	0.922	0.791	0.947	0.835	0.619
	GatedConv	—	—	—	0.973	0.914	0.767	0.953	0.849	0.621
	EdgeConnect	0.933	0.802	0.553	0.975	0.915	0.759	0.950	0.849	0.646
	PRVS	0.936	0.810	0.574	0.978	0.926	0.799	0.953	0.854	0.659
	RFR-Net(Ours)	**0.939**	**0.819**	**0.596**	**0.981**	**0.934**	**0.819**	**0.954**	**0.862**	**0.681**
PSNR*	PIC	27.14	21.72	17.17	30.67	24.74	19.29	29.35	23.97	19.52
	PConv	27.29	22.12	18.29	32.77	26.94	22.14	30.76	25.46	21.39
	GatedConv	—	—	—	32.56	26.72	21.47	31.32	25.54	20.61
	EdgeConnect	27.17	22.18	18.35	32.48	26.62	21.49	31.19	26.04	21.89
	PRVS	27.41	22.36	18.67	33.05	27.24	22.37	31.49	26.17	22.07
	RFR-Net(Ours)	**27.75**	**22.63**	**18.92**	**33.56**	**27.76**	**22.88**	**31.71**	**26.44**	**22.40**
Mean $l_1^\dagger$	PIC	0.0161	0.0441	0.0944	0.0111	0.0314	0.0749	0.0140	0.0379	0.0799
	PConv	0.0154	0.0409	0.0824	0.0083	0.0236	0.0524	0.0123	0.0313	0.0623
	GatedConv	—	—	—	0.0088	0.0245	0.0561	0.0120	0.0309	0.0660
	EdgeConnect	0.0157	0.0408	0.0821	0.0088	0.0247	0.0572	0.0110	0.0286	0.0582
	PRVS	0.0148	0.0390	0.0778	0.0079	0.0224	0.0500	0.0111	0.0281	0.0562
	RFR-Net(Ours)	**0.0142**	**0.0381**	**0.0761**	**0.0075**	**0.0212**	**0.0470**	**0.0110**	**0.0275**	**0.0546**

Table 1. Numerical comparison on three datasets. *Higher is better. †Lower is better.

Backbone	Coarse-To-Fine	PConv	bare RFR-Net
Model Size	100M+	33M	31M
SSIM	0.768	0.759	0.796
PSNR	23.10	23.69	24.60

Table 2. Model Size of different backbones and their quantitative performances on Paris StreetView dataset of 40%-50% mask.

Masked Image | Existing Attention 0.792/24.48/0.0388 | KCA 0.799/24.69/0.0375

Figure 7. Comparison results for different attention manners. From the left to the right are: (a) Input, (b) Existing Attention, (c) Knowledge Consistent Attention.

5.3. Ablation Studies

In this section, we would like to verify the effect of our contributions separately. Here we mainly illustrate the effectiveness of KCA module and the influences of the recurrence number ***IterNum***. Due to the space limitation, more ablation studies are placed in the supplementary materials, including 1) moving the RFR module 2) the effect of feature merge and 3) applying the RFR module in other models.

Effect of Knowledge Consistent Attention As mentioned in Section 3.2, directly using the attention module from [29] in RFR module results in images with some boundary artifacts, as shown in Fig. 7. This is because the inconsistency of the feature map might lead to shadow-like artifacts after the feature merging process. The SSIM, PSNR, and Mean l_1 below these images demonstrate the quantitative comparisons between attention modules. These numbers are calculated from all images on Paris StreetView given the mask ratio of 40%-50%.

IterNum	6	7	8
SSIM/PSNR	0.857/26.26	0.852/26.07	0.854/26.15

Table 3. The influences of different ***IterNum***s, the number indicates the chosen number of recurrences.

The effect of recurrence number *IterNum* The results on Paris dataset corresponding to different ***IterNum***s after the same training iterations are given in Table. 3. This ablation study reveals that our method is robust to the change of this hyper-parameter. The results also show that the improved performance compared to previous method is not from the deeper layers but from a more efficient architecture, since more ***IterNum***s do not improve the performance and our model has a smaller size than SOTAs.

6. Conclusion

In this paper, we propose the Recurrent Feature Reasoning network (RFR-Net) which progressively enriches information for the masked region and gives semantically explicit inpainted results. Further, a Knowledge Consistent Attention module is developed to assist the inference process of the RFR module. Extensive quantitative and qualitative comparisons, efficiency analysis, and ablation studies are conducted, which demonstrates the superiority of the proposed RFR-Net in both performance and efficiency.

7. Acknowlgement

This work was supported by the National Natural Science Foundation of China under grants 61822113 and 61771349, the Science and Technology Major Project of Hubei Province (Next-Generation AI Technologies) under Grant 2019AEA170, and Australian Research Council Project FL-170100117.

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM TOG*, 28(3):24, 2009. 1, 2

[2] Marcelo Bertalmio, Luminita Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *IEEE TIP*, 12(8):882–889, 2003. 2

[3] Tony F Chan and Jianhong Shen. Nontexture inpainting by curvature-driven diffusions. *JVCIR*, 12(4):436–449, 2001. 2

[4] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE TIP*, 13(9):1200–1212, 2004. 1

[5] Ding Ding, Sundaresh Ram, and Jeffrey J Rodríguez. Image inpainting using nonlocal texture matching and nonlinear filtering. *IEEE TIP*, 28(4):1705–1719, 2019. 2

[6] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei Efros. What makes paris look like paris? *ACM TOG*, 31(4):101, 2012. 6

[7] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks V.S. Lakshmanan, and Xuemin Lin. Efficient algorithms for densest subgraph discovery. In *Proc. VLDB*, pages 1719–1732, 2019. 2

[8] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014. 3

[9] Zongyu Guo, Zhibo Chen, Tao Yu, Jiale Chen, and Sen Liu. Progressive image inpainting with full-resolution residual network. In *Proc. ACM MM*, pages 85–100, 2019. 2, 3

[10] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM TOG*, 36(4):107, 2017. 3

[11] Jingyuan Li, Fengxinag He, Lefei Zhang, Bo Du, and Dacheng Tao. Progressive reconstruction of visual structure for image inpainting. In *Proc. ICCV*, pages 6721–6729, 2019. 2, 3, 6, 7

[12] Kangshun Li, Yunshan Wei, Zhen Yang, and Wenhua Wei. Image inpainting algorithm based on TV model and evolutionary algorithm. *Soft Comput.*, 20(3):885–893, 2016. 2

[13] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proc. ECCV*, pages 85–100, 2018. 3, 4, 6, 7

[14] Hongyu Liu, Bin Jiang, Yi Xiao, and Chao Yang. Coherent semantic attention for image inpainting. In *Proc. ICCV*, pages 4170–4179, 2019. 3, 7

[15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proc. ICCV*, pages 3730–3738, 2015. 6

[16] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. 3

[17] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Qureshi, and Mehran Ebrahimi. Edgeconnect: Structure guided image inpainting using edge prediction. In *Proc. ICCV Workshops*, 2019. 1, 3, 6

[18] Seoung Wug Oh, Sungho Lee, Joon-Young Lee, and Seon Joo Kim. Onion-peel networks for deep video completion. In *Proc. ICCV*, pages 4403–4412, 2019. 3

[19] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proc. CVPR*, pages 2536–2544, 2016. 3

[20] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *Proc. SIGGRAPH*, pages 313–318, 2003. 3

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241, 2015. 7

[22] Rakshith Shetty, Mario Fritz, and Bernt Schiele. Adversarial scene editing: Automatic object removal from weak supervision. In *Proc. NeurIPS*, pages 7717–7727, 2018. 1

[23] Linsen Song, Jie Cao, Linxiao Song, Yibo Hu, and Ran He. Geometry-aware face completion and editing. *CoRR*, 2018. 1, 2

[24] Ning Wang, Jingyuan Li, Lefei Zhang, and Bo Du. Musical: Multi-scale image contextual attention learning for inpainting. In *Proc. IJCAI*, pages 3748–3754, 2019. 3

[25] Chaohao Xie, Shaohui Liu, Chao Li, Ming-Ming Cheng, Wangmeng Zuo, Xiao Liu, Shilei Wen, and Errui Ding. Image inpainting with learnable bidirectional attention maps. In *Proc. ICCV*, pages 8858–8867, 2019. 1, 3

[26] Wei Xiong, Jiahui Yu, Zhe Lin, Jimei Yang, Xin Lu, Connelly Barnes, and Jiebo Luo. Foreground-aware image inpainting. In *Proc. CVPR*, 2019. 3

[27] Zhaoyi Yan, Xiaoming Li, Mu Li, Wangmeng Zuo, and Shiguang Shan. Shift-net: Image inpainting via deep feature rearrangement. In *Proc. ECCV*, pages 3–19, 2018. 1, 3

[28] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proc. CVPR*, pages 5485–5493, 2017. 3

[29] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proc. CVPR*, pages 5505–5514, 2018. 1, 2, 3, 4, 8

[30] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Free-form image inpainting with gated convolution. In *Proc. ICCV*, pages 4471–4480, 2019. 1, 3, 6, 7

[31] Haoran Zhang, Zhenzhen Hu, Changzhi Luo, Wangmeng Zuo, and Meng Wang. Semantic image inpainting with progressive generative networks. In *Proc. ACM MM*, pages 770–778, 2018. 2, 3

[32] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Pluralistic image completion. In *Proc. CVPR*, pages 1438–1447, 2019. 6

[33] Bolei Zhou, Àgata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE TPAMI*, 40(6):1452–1464, 2018. 6

Supplementary Material

A. Network Architecture

The detailed architecture of our model is shown in Table 4. The upper half of the table is the architecture of the whole RFR-Net. The bottom half is the architecture of the RFR Module. In specific, the meanings of the table are in the first row. Input_Feat tells the source of the feature. In_Size and Out_Size indicate the sizes of the feature maps after they are processed and Ori means the size of the original input. K_Size means the kernel size of the operators. Stride means the stride of the operators, which will be omitted if this parameter is not applicable. Num_Chan means the channel number of the output feature map. BN means whether batch normalization layer is used after the operator. Act_Fun means the non-linear function after the layer. Note that the RFR module actually has a recurrent design that feeds the feature map from layer "Deconv3" into "PartialConv1". Except for the first time recurrence, the input mask for "PartialConv1" comes from "PartialConv2" in the last recurrence. All leaky relu layers have a negative slope of 0.2.

Connection to Partial Convolution U-Net: The key component of the RFR-Net is the RFR module with a recurrent design that progressively recovers the damaged deep feature maps. To detect the area to be processed in next recurrence for RFR, there are several options, such as "Partial Convolution", "Gated Convolution" and "Learnable Attention Maps". For simplicity, we used the layer and loss functions from Partial Convolution U-Net. However, the inpainting process of our RFR-Net differs largely from Partial Convolution U-Net. First, RFR-Net aims to generate high quality features in each recurrence so that the subsequent recurrences can benefit from them while the architecture from Partial Convolution U-Net propagates contents to the holes aggressively in each convolution layer, leading to information distortion during the process. Second, RFR-Net deploys shared parameters in each recurrence under a recurrent architecture while Partial Convolution U-Net calculates each step with different parameters under a feedforward architecture. Third, RFR-Net merges features generated from different recurrences to stabilize backward propagation by eliminating the gradient vanishing problem, while Partial

RFR-Net Architecture								
Module Name	Input_Feat	In_Size	K_Size	Stride	Num_Chan	Out_Size	BN	Act_Func
PartialConv0	Img_Masked	Ori	7	2	64	Ori/2	T	ReLU
PartialConv1	F_Pconv0	Ori/2	7	1	64	Ori/2	T	ReLU
RFR Module	F_Pconv1	Ori/2			64	Ori/2	F	None
DeConv4	F_RFR	Ori/2	4	2	64	Ori	T	Leaky_ReLU
PartialConv4	Cat(Img_Masked, F_Deconv4)	Ori	3	1	32	Ori	F	Leaky_ReLU
Conv9	F_Pconv3	Ori	3	1	32	Ori	T	Leaky_ReLU
Con10	F_Conv9	Ori	3	1	32	Ori	T	Leaky_ReLU
Output_Conv	Cat(F_Pconv3, F_Conv11)	Ori	3	1	3	Ori	F	None
RFR Module Architecture								
Module Name	Input_Feat	In_Size	K_Size	Stride	Num_Chan	Out_Size	BN	Act_Func
PartialConv2	F_Pconv1	Ori/2	7	1	64	Ori/2	F	None
PartialConv3	F_Pconv2	Ori/2	7	1	64	Ori/2	T	ReLU
Conv1	F_Pconv3	Ori/2	3	2	128	Ori/4	T	ReLU
Conv2	F_Conv1	Ori/4	3	2	256	Ori/8	T	ReLU
Conv3	F_Conv2	Ori/8	3	2	512	Ori/16	T	ReLU
Conv4	F_Conv3	Ori/16	3	1	512	Ori/16	T	ReLU
Conv5	F_Conv4	Ori/16	3	1	512	Ori/16	T	ReLU
Conv6	F_Conv5	Ori/16	3	1	512	Ori/16	T	ReLU
Conv7	Cat(F_Conv6, F_Conv5)	Ori/16	3	1	512	Ori/16	T	Leaky_ReLU
Conv8	Cat(F_Conv7, F_Conv4)	Ori/16	3	1	512	Ori/16	T	Leaky_ReLU
KCA	F_Conv8	Ori/16			512	Ori/16	F	None
DeConv1	Cat(F_KCA, F_Conv3)	Ori/16	4	2	256	Ori/8	T	Leaky_ReLU
DeConv2	Cat(F_Deconv1, F_Conv2)	Ori/8	4	2	128	Ori/4	T	Leaky_ReLU
DeConv3	Cat(F_Deconv2, F_Conv1)	Ori/4	4	2	64	Ori/2	T	Leaky_ReLU
Feature Merge	All F_Deconv3	Ori/2			64	Ori/2	F	None

Table 4. The architecture of the RFR-Net and RFR module respectively.

Method	SSIM*			PSNR*			Mean $l_1{}^\dagger$			Memory Usage
Mask Ratio	0.1-0.2	0.3-0.4	0.5-0.6	0.1-0.2	0.3-0.4	0.5-0.6	0.1-0.2	0.3-0.4	0.5-0.6	ANY
DownSamp 0	Unable to train			Unable to train			Unable to train			1657M
DownSamp 1	0.955	0.860	0.673	31.72	26.38	22.29	0.0110	0.0279	0.0558	1122M
DownSamp 2	0.949	0.845	0.650	30.99	25.78	21.86	0.0118	0.0296	0.0582	863M
DownSamp 3	0.948	0.839	0.635	30.93	25.64	21.70	0.0129	0.0303	0.0600	834M

Table 5. Comparison between different places to put the module. *Higher is better. †Lower is better.

Masked Image | No Merge | Average Merge | Adaptive Merge

Figure 8. Different methods for feature merging in RFR module. From the left to the right are: (a) Input, (b) No Merging, (c) Average Merging, (d) Adaptive Merging.

Convolution U-Net does not have the gradient vanishing problem. Finally, we designed KCA to search for feature globally while Partial Convolution U-Net only captures information locally.

The design of Feature Reasoning module: The module aims to partly recover the masked region in deep feature maps. Therefore, we need to extract information from the already known features and estimate new contents. In this module, such new contents are produced in a feature reconstruction style with additional consideration. According to this objective, we develop our architecture based on the standard feature extraction and reconstruction technique, i.e., the encoder-and-decoder design.

Computational Complexity: The recurrent design of our RFR module increases the computational complexity compared to its non-recurrent version. However, since the RFR module is implemented for down-sampled feature maps (with much smaller size), the additional computation cost for each extra ***IterNum*** during inference is very limited ($\sim$8ms and $\sim$20mb for each ***IterNum***).

B. More Ablation Studies

In this section of the supplementary material, we will show more ablation study about the RFR module. In specific, we first show that the RFR module can be installed in any part of an existing network and the computational cost can be controlled. Then we show that the RFR module can benefit a network whose input and output is not represented in the same space.

B.1. Moving the RFR Module

In this section, we will test how will the network's performance change if we move the RFR module up and down in the network.

In specific, we will move the RFR module up and down in the network by adding or modifying the encoding and decoding layers to show that the RFR module can be flexibly installed into any part of a network. We tested four different models, which are using the RFR without downsampling layers (Modifying "PartialConv0" and "DeConv4" in the Table 4), using the RFR after downsampling for once (original design in Table 4), using the RFR after downsampling for twice (adding extra encoding and decoding layers before and after the RFR module respectively) and using the RFR after downsampling for three times. The RFR modules for each experiment remain the same to address any possible influence, which means the input feature maps all have 64 channels. Attention module is removed from all models we tested here. All results are from models trained on Paris StreetView dataset. For no-downsampling case, we are not able to train the network due to its unacceptably high computational cost and we only conduct memory cost experiment. By analyzing the Table 5, we notice that by moving the RFR module deeper, the computational cost is significantly reduced while the performances are only affected little. This further shows the superiority of the RFR module and the potential of the RFR-Net.

B.2. Effect of Feature Merging

Fig. 8 compares of different feature merging approaches in the RFR module on Paris StreetView. If only the last feature map is used as output (Fig. 8 (b)) for feature merging, the texture is blurred and inadequate. This is because during the feature reasoning process, some feature generated in earlier recurrences might be damaged in order to further recover the hole region. Further, when we replace our adaptive merging (Fig. 8 (d)) with average merging (Fig. 8 (c)), the restored details of average merging are worse than those of adaptive merging. The reason is that when performing average merging, the feature values in the hole region are smoothed partly, as we explained in Sec. 3.1.3. This part demonstrates the advantage of the adaptive feature merging scheme we proposed.

B.3. RFR Module For Structure Estimation

In this section, we will show that the RFR module can also boost the performance of a multi-stage method's subnetwork. In specific, the multi-stage method we choose is Edge-Connect, which first uses a network to reconstruct the boundary from the corrupted image and use the boundary to guide image inpainting. In this case, the each single network has different input and output space and therefore all existing progressive methods are not feasible. We replace the 8 residual blocks in the first network of the model with the RFR module and compare the results. The training settings are kept the same as that in the original paper of Edge-Connect. In Fig. 9, (a) and (d) are the masked input images. (b) and (e) are the results from Edge-Connect method's structure generator. (c) and (f) are the results from the RFR structure generator. The results from the RFR module are significantly better than the original edge generator. The results from this section also demonstrates the potential applications of the RFR net on other tasks where the input and output are not in the same representation space and some parts of the network include a encoder-decoder design architecture.

(a) (b) (c) (d) (e) (f)

Figure 9. RFR module for structure estimation.

C. More Results

In this part, more visual comparisons and results are exhibited. In the first part, visual comparisons with state-of-the-art methods on CelebA and Paris StreetView datasets, which are omitted in the main text of the paper due to the space limitation. Then we show more visual results on three datasets with ground truth images. All these results demonstrates the effectiveness of our proposed methods.

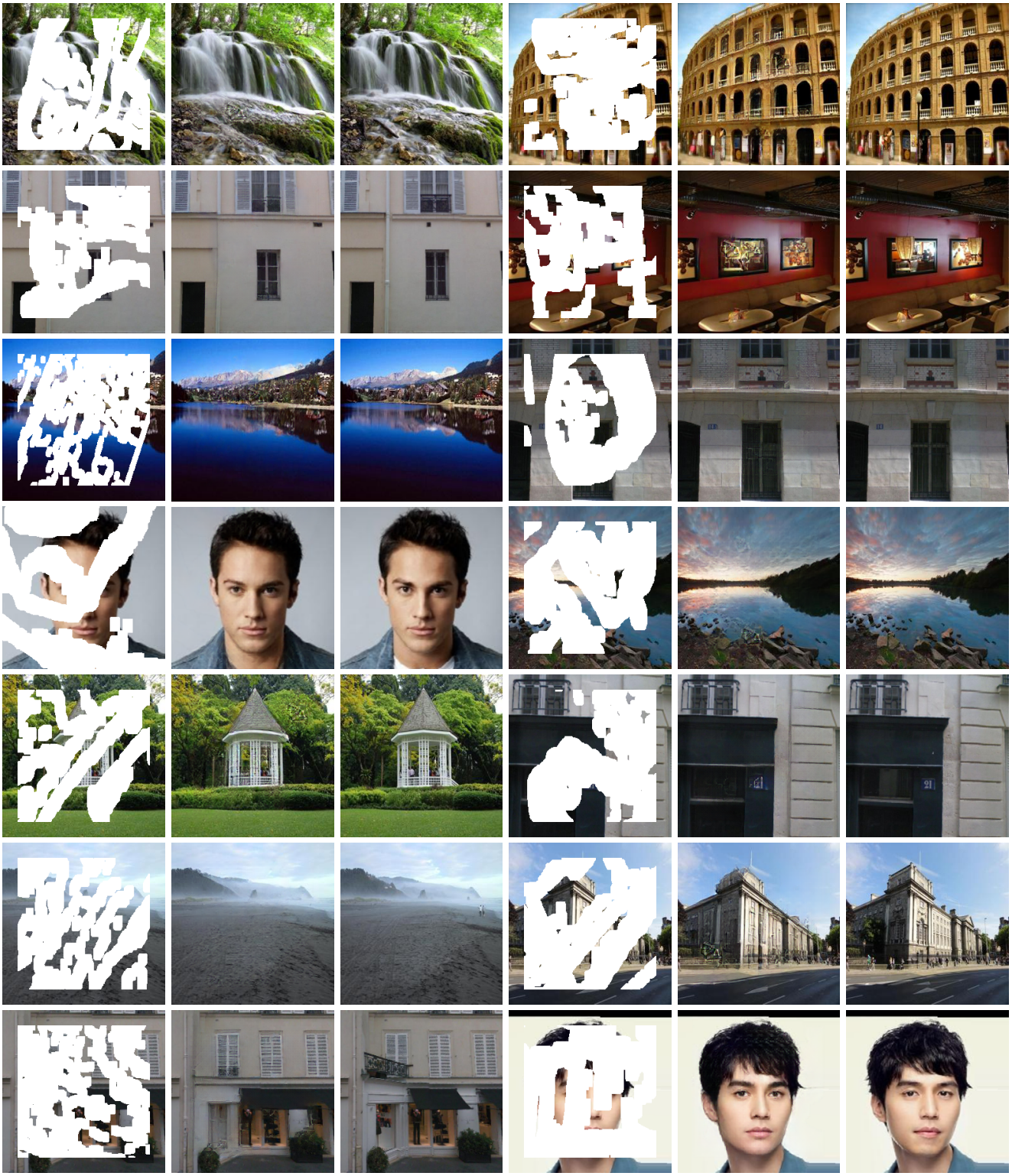
C.1. More Comparisons

In this section, we show more comparison results. The results are on Paris StreetView and CelebA datasets. Our RFR-Net exhibits less boundary artifacts and much more explicit generated content.

(a) (b) (c) (d) (e) (f)

Figure 10. More comparison results on Paris Street View and CelebA datasets. Our model stably produces well-structured results, even if the holes are large and challenging. From the left to the right are: (a) Input, (b) Ground Truth, (c) GatedConv, (d) PConv, (e) EdgeConnect and (f) Our RFR-Net.

C.2. More Visual Results



(a) (b) (c) (d) (e) (f)

Figure 11. More visual results. Our results have both coherent structures and plausible details. From the left to the right are: (a) Input, (b) Our RFR-Net, (c) Ground Truth, (d) Input, (e) Our RFR-Net and (f) Ground Truth.