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	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.

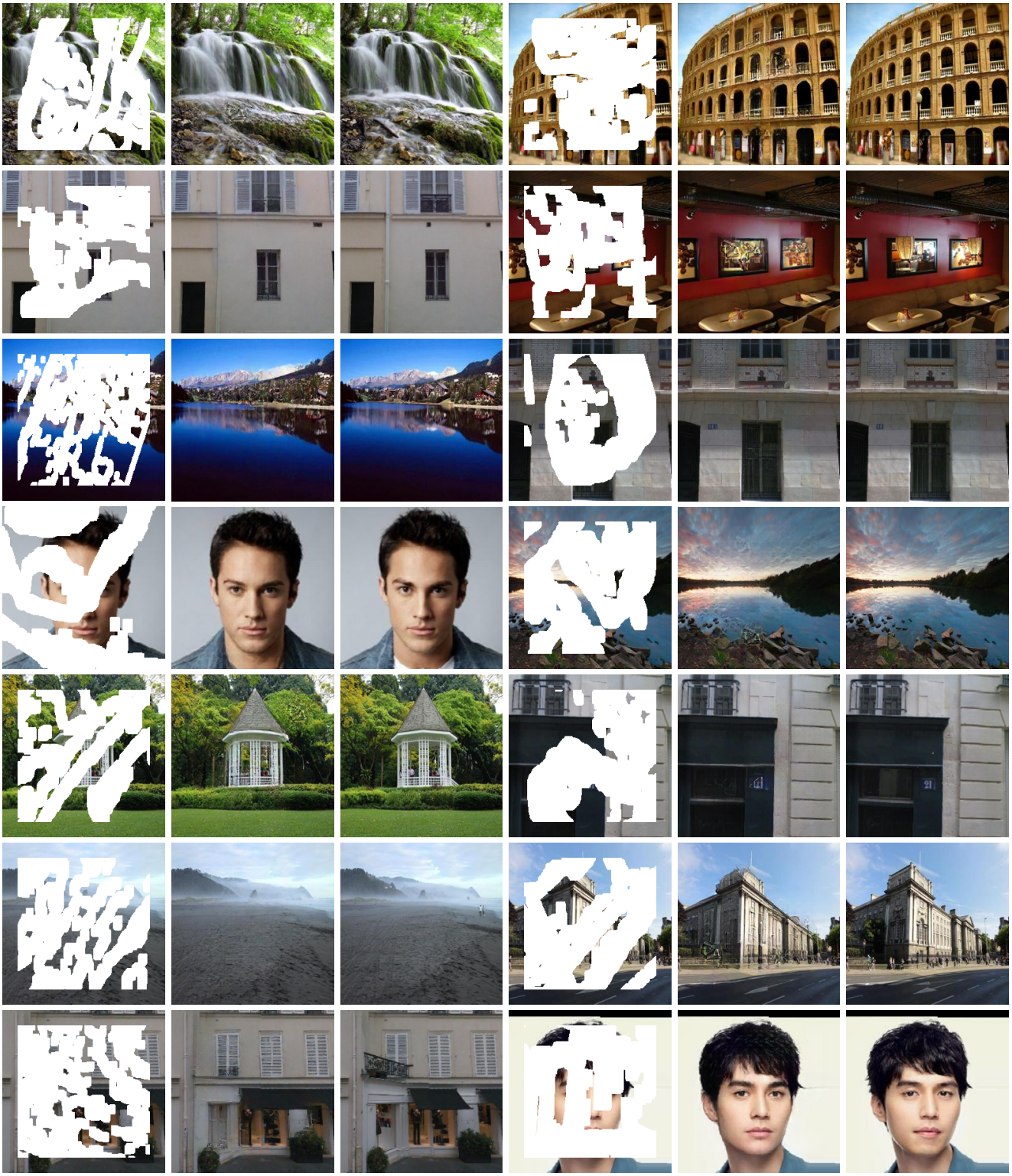
C.1. More Comparisons

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.