Restoration of Partially Pixelated Image

JunYoung Park, DongHyun Kwon
Department of Computer Science and Engineering, POSTECH
{parkjuny, kinux98}@postech.ac.kr

Abstract

There had been many proposed methods of obtaining super-resolution (SR) image from low resolution image. The problem of restoring pixelated part of image is similar with SR, but it is different from SR in many ways so methods of SR can't be simply applied in restoration of partially pixelated images. However, it doesn't mean the methods of SR can't be used in restoring pixelated part of the image, so we designed our model using residual learning [5] and recurrent network architecture, so that information of non-pixelated part of the input image are used in restoration of pixelated part of input image step by step, and non-pixelated part of the image comes out as output nearly identical to part of input image. Our experiments show that if the surrounding image is badly damaged, the restoration level of our model is low, but if there are some guessable informations, our model restores the image to a meaningful level. Our github link is here.

1. Introduction

For a few recent years, there have been many attempts to restore the high resolution images from low resolution images. The problem of making high resolution images from low resolution images is ill-posed since multiple HR images may result in a single LR image. To solve this problem, many super resolution methods have been proposed.

Restoration problem of pixelated part of image is similar with the super resolution since pixelated part of image can be considered as LR image because the process of pixelation is basically downscaling and upscaling the image. Because of the similarity of two problems, methods of super resolution can be used in restoration of fully pixelated images. However, restoration of partially pixelated images is different from SR since the part of image that is not pixelated is identical to the part of original image, and resolutions of original image and partially pixelated image are same unlike the resolutions of input image and output image of SR are different. However, it doesn't mean that methods of SR can't be used in restoring partially pixelated images since there are many commonly usable methods in SR such as residual learning [5], feedback network [6] .

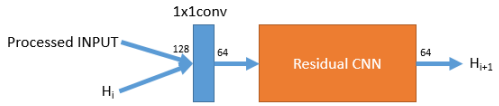

Figure 1. The illustration of hidden state H_i augmented with processed input. They are combined with 1x1 convolution layer, and works as an input of Residual CNN.

In this report, we designed a network for restoration of partially pixelated images, which has a recurrent structure with a residual CNN (Convolutional Neural Network) [2] inside. Inspired by [6] , we designed our network to have recurrent connection, so that the designed network is a RNN with a residual CNN inside, so that the input image passes through residual CNN multiple times. The output image becomes clearer and clearer as it passes through residual CNN more. If we unfold our network, the hidden state, which is the output of residual CNN, flows into the next iteration and modulated as an input of the residual CNN of next iteration. (see Fig. 1) Hidden states are augmented with input of the model, so the part of the image that is not pixelated becomes augmented with hidden states. Thus, having a recurrent structure helps the information of non-pixelated part of image to be used in restoring the pixelated part of the image. The output loss of the model is calculated as summation of losses of output images, to ensure that the hidden states of our model contain the information of original image.

In summary, we designed a network in a recurrent network structure with a residual CNN inside. Recurrent network structure helps the information of the part of image that is not pixelated to be efficiently used in restoring the pixelated part of the image.

2. Related Works

2.1. Residual Learning

If the neural net's learning objective is to find a function $H(x)$ that maps the input x to the target value y, the neural

net learns to minimize $H(x) - y$ In this case, y paired with x actually represents x, especially in the image classification problem, so that the input and output of the network should be mapped semantically. So in ResNet, it changed our perspective by changing network to get $H(x) - x$ for the network. Define the residuals of input and output as $F(x) = H(x) + x$ and the network finds this $F(x)$. This $F(x)$ can be referred to as a residual, and learning such a residual is called residual learning [5] or residual mapping. As a result, the output becomes $H(x) = F(x) + x$. This use of the network inputs and outputs as the input to the next layer is called a skip connection. If the original neural net was to make $H(x)$ equal to the correct answer somehow, it should now be to learn the residual between input and output, so the learning goal is already set and the learning speed will be faster, and the network will Once learned, it will be sensitive to small changes in input.

2.2. RNN

RNN is a type of artificial neural network in which hidden nodes are connected by directional edges to form a direct cycle. It is known as a model suitable for processing data that appears sequentially in voice and text. Along with Convolutional Neural Networks (CNN), this algorithm has been in the spotlight recently. RNN has been successfully applied in many natural language processing problems. Currently, LSTM [3] is the most widely used type of RNN in the world because it effectively remembers longer sequences than the basic RNN structure discussed above. We decided to apply the pixelate image recovering inspired by the RNN structure.

2.3. SRFBN

Recent advances in ultra high resolution (SR) of images have been worked to explore the power of deep learning to achieve better reconstruction performance. However, the feedback mechanisms commonly present in the human visual system are not fully utilized in traditional deep learning based image SR methods. The feedback mechanism allows the network to know the concept of output to solve the previous state. Recently, feedback mechanisms are used in many network architectures for various visual tasks.

The feedback system contains two requirements: repeatability and output rerouting of the system to modify the input of each loop. This iterative process of cause-and-effect helps to achieve the principles of the image SR feedback scheme: Higher level information can guide LR images and better SR image recovery. There are three essential parts to conducting a feedback scheme in the proposed network: linking losses at each iteration (to rebuild the SR image at each iteration to the network and make it possible to carry a high level information concept with hidden state), [6] in order to use the iteration structure (to realize the iterative process), and RNN to provide the LR input at each iteration (to ensure the availability of low-level information need to be refined). Without these three parts, the network cannot drive the feedback flow.

2.4. VDSR

VDSR(Very Deep Convolutional Networks) [5] is a super-resolution model inspired by VGG-net which is used for ImageNet classfication. It has Convolution-Relu block inside between input and output layers. VDSR uses a deep CNN (20 weight layer) inspired by vgg-net used for imagenet classification. It is possible to extract meaningful information efficiently while passing many small filters continuously in a deep network. In addition, VDSR solved the complex computational problem of deep network by residual learning. The training speed could be increased by increasing the learning rate. What's special about this model is that it puts the y component of the image as input. This model is intended to solve a typical SR problem but has been found to not produce very good results for recovering pixelated images.

3. Designed Model

3.1. Characteristics of Problem

Restoration of partially pixelated images has several characteristics. First, the part of the input image which is not pixelated should come out identically in the output. Since the input image is partially pixelated, it has identical parts with output image, and those parts should come out identically. Second, the lost information of pixelated part of input image may be restored from parts of the image that is not pixelated. For example, if features like lines or circles are cut due to pixelation, they might be restored from parts that are not pixelated, by predicting them as lines or circles. Lastly, the resolution of input image and output image is same, so most of the methods of super resolution can't be used in restoring partially pixelated images.

3.2. Network Structure

Our designed model is based on RNN, with a residual CNN inside. (see Fig. 2) We designed our model to have recurrent structure because having a recurrent structure helps the information of the part of image that is not pixelated to be used in restoring the pixelated part of the image. We implemented our model to have T recurrent steps, which was based on limitation of our graphics memory, so that T images come out as output. Our model also have direct connection from input to output for residual learning since residual learning strategy is more ensuring for non-pixelated parts to come out in output nearly identically.

We used ReLU as the activation function of our network, which was applied after convolution layers every time ex-

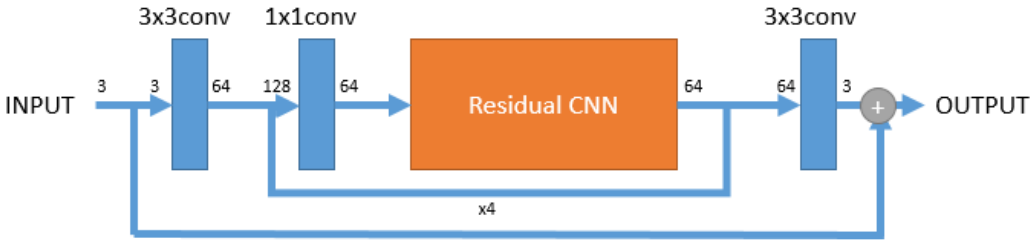

Figure 2. Architecture of our model. The figure of residual CNN is in Figure 3.

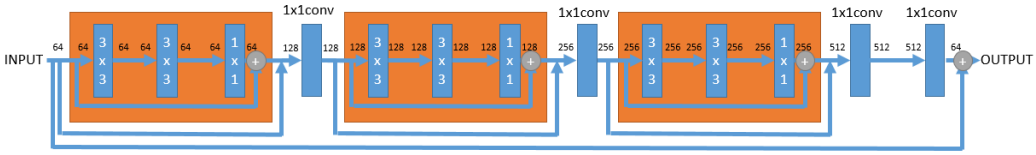

Figure 3. Architecture of Residual CNN.

cept the last 3x3 convolution layer.

3.3. Feedback Mechanism

Our designed model can be unfolded to T iterations. Hidden states H_i can be defined as an output of residual CNN in iteration i. Hidden state H_i is stored inside the model and is used in iteration $i+1$. Hidden state H_i is augmented with the processed input in 1x1 convolution layer, and works as an input of residual CNN in iteration $i+1$. (see Fig. 1) For the first iteration, H_0 is initialized with zero tensor.

As iteration passes, hidden states contain more information of non-pixelated part of input image. Also, the output image gets clearer as it passes through residual CNN several times. The information of non-pixelated part of input image becomes augmented with hidden states, and this information is used to restore pixelated part of input image.

Hidden states of all iterations should obtain the information of original image. For that, loss function is designed as a summation of loss values of output images.

$$L(\theta) = \sum_{i=1}^{T} L(I_{out}, I_{orig}) \tag{1}$$

If we perform back propagation on that loss function, back propagation will be performed for output images of all iterations, which allows hidden images to capture information of original image.

4. Experimental Results

4.1. Settings

Training Dataset We created training dataset based on DIV2K dataset [1], by randomly cropping the image by 64x64, randomly select area of cropped image, resize it by $1/10$ and scale it up by 10 to make the size back, put the modified image back to the cropped image so that we can get partially pixelated image. (see Fig. 4) 4 partially pixelated images are generated in this process because there are 4 modes of resizing images in OpenCV2, which we used to generate partially pixelated cropped images. With DIV2K train dataset, we generated 64 randomly cropped images per image in DIV2K train dataset, and we generated 4 partially pixelated images from them so we could make our dataset with size 204800.

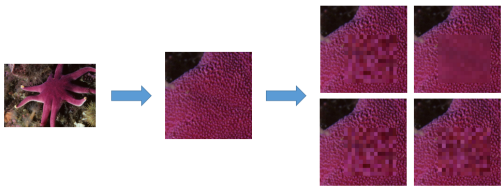

Figure 4. The illustration of making partially pixelated images. Randomly crop image with 64x64 size, randomly select the area, scale down and scale up the selected area with four modes. (linear, cubic, area, nearest) Put the modified selected area back into the cropped image, so we get 4 partially pixelated images.

Validation Dataset We created validation dataset based on Set5 dataset [4], generating partially pixelated images with the same method of generating training dataset, but we didn't crop the image and ran the code until interesting points such as eyes or patterns are chosen to be pixelated. We made our dataset with size 20 with this method.

Metrics We used L1 loss function to train our model. Our model gives 4 images as output since our model have $T = 4$ recurrent steps, so summation of L1 losses is used

to train the model, ensuring hidden states of our model have information of original image. To evaluate our model, we used PSNR and SSIM [7] metrics on our final output after 4 recurrence steps.

Training We trained our model total 9 epochs, with learning rate 10^{-4}. (see Table. 1)

epoch	train loss	val loss	PSNR	SSIM
1	24.44488	7.317488	25.65992	0.830908
2	17.6947	6.362999	26.72091	0.854103
3	14.97461	5.989555	26.8865	0.860435
4	13.78725	5.793819	27.04678	0.862397
5	13.17438	5.715679	27.03616	0.863508
6	12.80694	5.67652	27.10314	0.864766
7	12.54943	5.698876	27.00887	0.864814
8	12.36891	5.633733	27.15549	0.865214
9	12.21555	5.628301	27.16065	0.865519

Table 1. The table shows the change of train loss, val loss, PSNR and SSIM according to epoch. The learning time per epoch is approximately an hour. PSNR and SSIM are the average values of the total restoration (20 images, 4 pixelated images per one HR image) of Set5 dataset.

4.2. Comparing with VDSR

VDSR is a sophisticated model built for SR. However, it has not been confirmed that it works well in restoring partially pixelated images. We decided to compare the results with our model and VDSR. We used pretrained VDSR model in Link. The test images are available at Link.

	Our Model		VDSR	
	PSNR	SSIM	PSNR	SSIM
baby_pixelated1	34.08	0.9344	28.90	0.7965
baby_pixelated2	33.36	0.9331	30.10	0.8068
baby_pixelated3	34.08	0.9336	28.64	0.7934
baby_pixelated4	33.36	0.9217	26.23	0.7771
bird_pixelated1	30.39	0.9769	29.90	0.8355
bird_pixelated2	34.41	0.9766	30.74	0.8379
bird_pixelated3	35.06	0.9767	29.80	0.8349
bird_pixelated4	31.87	0.9665	27.17	0.8221
buttefly_pixelated1	19.97	0.7984	17.07	0.7467
buttefly_pixelated2	21.08	0.7937	19.42	0.7554
buttefly_pixelated3	19.82	0.7977	16.82	0.7444
buttefly_pixelated4	18.72	0.7732	15.65	0.7188
face_pixelated1	31.03	0.9285	27.23	0.8376
face_pixelated2	32.33	0.9283	28.79	0.8448
face_pixelated3	30.71	0.9279	26.87	0.8347
face_pixelated4	27.98	0.9069	24.41	0.8098

Table 2. The table shows scores of two metrics: PSNR, SSIM. Results of left side is our model's, Right side shows VDSR's ones.

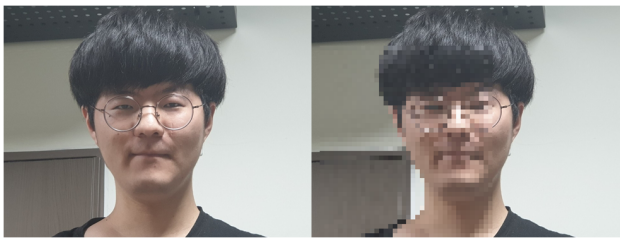

Figure 5. Left image is the original image, and right image is partially pixelated image.

Figure 6. Left image is our model's result. PSNR : 34.08, SSIM : 0.9344. Right image is VDSR's result. PSNR : 28.90, SSIM : 0.7965

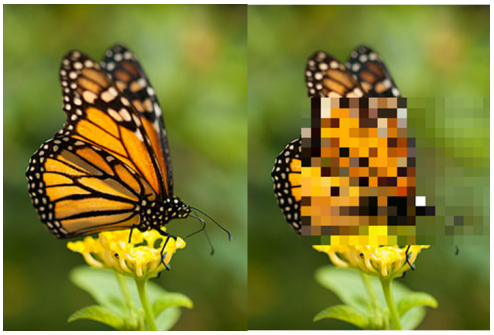

Figure 7. Left image is the original image, and right image is partially pixelated image.

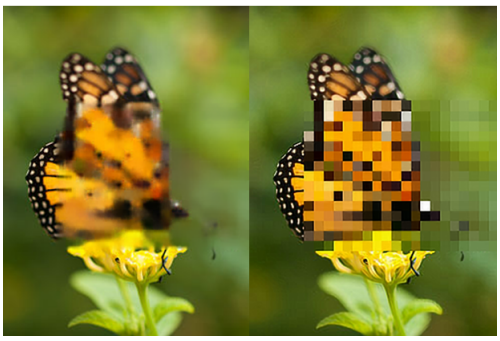

Figure 8. Left image is our model's result. PSNR : 19.97, SSIM : 0.7984. Right image is VDSR's result. PSNR : 17.07, SSIM : 0.7467

5. Conclusion

In this paper, we designed a model that utilizes RNN network for the purpose of restoring a partially pixelated image and focus on reconstruction from the pixelated image to original one. The recurrent structure provides information on the non-pixelated portion of the image, consequently helping to improve the pixelated portion. As a result, various verifications show that if the surrounding image is badly damaged, the degree of recovery is low, but if there are some guessable informations, the image is recovered to a meaningful level.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[4] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5197–5206, 2015.

[5] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR Oral)*, June 2016.

[6] Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu. Feedback network for image super-resolution. *CoRR*, abs/1903.09814, 2019.

[7] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 13(4):600–612, 2004.