Content-adaptive Representation Learning for Fast Image Super-resolution

Yukai Shi *, Jinhui Qin *

Abstract—Deep convolutional networks have attracted great attention in image restoration and enhancement. Generally, restoration quality has been improved by building more and more convolutional block. However, these methods mostly learn a specific model to handle all images and ignore difficulty diversity. In other words, an area in the image with high frequency tend to lose more information during compressing while an area with low frequency tends to lose less. In this article, we address the efficiency issue in image SR by incorporating a patch-wise rolling network(PRN) to content-adaptively recover images according to difficulty levels. In contrast to existing studies that ignore difficulty diversity, we adopt different stage of a neural network to perform image restoration. In addition, we propose a rolling strategy that utilizes the parameters of each stage more flexible. Extensive experiments demonstrate that our model not only shows a significant acceleration but also maintain state-of-the-art performance.

Index Terms—Image Super Resolution, Convolution Neural Network, Acceleration

I. INTRODUCTION

Deep learning has successfully applied in many computer vision fields such as image recognition [16], semantic segmentation [30] and object detection [26]. Inspired by the rapid development and superior performance, many efforts have been made to introduce deep learning in low-level vision as well as image processing tasks, including image super-resolution [5], image enhancement [15], inpainting [29] etc. Meanwhile, Single image super-resolution(SISR), namely to predict high-resolution with low-resolution input, is widely used in many computer vision applications and draws plenty of attentions [4], [5], [18], [19], [22], [24], [34].

Recently, Convolutional neural networks(CNNs) achieve magnificent improvement toward image restoration by adopting a building block strategy. VDSR [18] utilizes residual connection and a very deep model to achieve promising results in image SR. EDSR [24] further improves the results by adopting residual block [16] and remove batch normalization. However, they advance performance with numerous parameter gain and huge computational cost. Dense block [12] also exhibits its effectiveness in image enhancement. MemNet [37] realizes a coarse-to-fine restoration process by using dense block and recursive unit. Zhang et al. [45] proposes an optimized block, which combines the strengths of the dense block and residual block, and achieve impressive promotion. However, deep learning-based SR methods [18], [24], [37], [45] prefer to crop the image into patches before training phrase. As different patch has various texture and structure, it is inefficient to adopt a feed-forward network to super-resolve all samples, especially for those intensely simple patches. In addition, notwithstanding such a complicated model can bring positive performance with a graphics processing unit(GPU), it also leads to expensive computational cost and explosion of parameters.

Computer vision applications and technogies [10], [15], [44] for mobile devices draw a lot of attention as it has wide application scenarios. However, using CNNs on the mobile platform has an extreme requirement towards efficiency. MobileNet [10] makes an attempt to accelerate speed by utilizing a depth-wise convolution to reduce redundancy of CNNs. Similar technology also adopted by ShuffleNet [44]. Moreover, ShuffleNet employs a novel shuffle unit, which maintains performance with efficiency improvement. However, their methods are limited by the optimization of the computational platform and sometimes run inefficiently. IGC [43] utilizes parameters of the deep network more efficiently by adopting group convolution and permutation of convolutional features. The similar idea also used by RRC [28]. RRC implements a rolling strategy on object detection, which not only utilizes multi-scale features but also realizes an efficient one-stage framework. Their methods reveal that features of different scale can be utilized more efficiently. MSDNet [11] proposes a multi-scale dense net, which adaptively uses the specific stage in the deep model to deal with samples with
different difficulty levels. For instance, MSDNet adopts early stage convolutional layers to handle easy samples and more parameters are applied to process difficult images. However, MSDNet can inherently distinguish difficult level with an internal high-level representation of the image itself. Since such internal high-level prior is not exist in low-level vision, MSDNet is fail to applied in images processing tasks.

Motivated by previous works, we make an attempt to propose a content-adaptive and flexible framework, which can accurately super-resolve image with different difficulty level according to gradient prior. In the proposed model, we first define the gradient prior to distinguish different samples. Then, a unified model is proposed to handle samples with different difficulty by a content-adaptive fashion. Since samples with different difficulty will cause frequency conflicts and result in a performance degradation. We also propose a flexible rolling strategy by alternating the convolution filters to address this problem.

Our main contributions are summarized as follows.

- We find it is inefficient to adopt an expensive model to mild samples, which have less texture and simple structure. In contrast, an expensive model is appropriate for the samples, which have rich texture and complicated structure.
- According above observation, we distinguish the difficulty of samples by its gradient prior and content-adaptively adopt different convolutional stage to super-resolve samples. This strategy helps us greatly improve SR efficiency.
- Since the samples with different difficulty exhibit various property in the frequency domain, which causes frequency conflicts and leads to a performance degradation. We propose a flexible rolling strategy. With our rolling approach, our model not only achieve a balance between mild and severe samples but also increase the receptive field of early layers.

II. RELATED WORK

CNN for image SR. Recently, deep learning based SR methods have achieved a great successes in many computer vision fields. Super-resolution, which considered a typical low-level vision task and is well-known for its ill-posed property, plays an important role in image quality enhancement. Many researchers devote themselves to the studies of super-resolution and have proposed many insightful works. Recently, the rising of deep learning methods give new solution to image SR. Dong et al. [5] first adopt deep convolutional neural networks to learn the mapping from LR to HR patches in an end-to-end manner and greatly boost the performance of image SR. Afterward, many deep learning based methods have been proposed to improve the performance mainly by developing the network architecture. VDSR [18] and IRCNN [42] increased the network depth by adding more convolutional layers, and DRCN [17] introduced recursive learning for parameter sharing. Tai et al. introduced recursive blocks in DRRN [36] and memory block in Memnet [37]. While all of these methods have greatly improved the SR performance by exploiting different network architecture, they have not considered the efficiency of SR, which lead to the learning based SR methods been away from application in reality.

In contrast to chasing a smaller mean square error, we focus on the improvement of image restoration quality as well as boost the speed of the algorithm, which has been neglected for a long time. FSRCNN [4] make an attempt to address this issue by adopting down-sampled patches as input and deconvolution to speed up the computing process. Their method effectively reduce redundancy and inspired us to explore the potential of accelerating SR. ESPCN [35] used pixel shuffling operation to reduce features volume and checkerboard effect, which also greatly accelerated the SR network. Although these methods obtain a small running time, they don’t fully utilize the inherent property of SR problem. For image SR, it has internal difficulty diversity, that is an area of an image with high frequency tend to lose more information during compressing while an area with low frequency tends to lose less. However, aforementioned methods ignore this property and tend to adopt a feed-forward model to process all samples.

Neural network acceleration. Obtaining a better balance between accuracy and efficiency has attracted many research communities for decades. Many studies have been proposed to change the connectivity structure of the deep convolutional networks such as ShuffleNet [44] or introduce a more compact convolution operation such as in MobileNet [10] and MobileNetV2 [31]. These studies have done great in reducing computation cost as well as maintain or even improve performance. However, these methods can be slower than a plain network in some computing platform. Some studies focus on reducing model size after training, such as weight pruning [21], [23], weight quantization [14], [27]. These studies construct new models at the test time and re-train or fine-tune them to achieve a similar closer performance as the original models.

Other studies focus on alternating the evaluation manner. FractalNets [20] perform prediction at any time by progressively evaluating subnetworks of the full network. Bolukbasi et al. [2] addresses this problem by adaptively evaluating neural networks. Different from these works, MSDNet [11] adopts a specially designed network with multiple classifiers, which can directly output confidence scores to control the evaluation process for each test example. The adaptive computation time method [9] and its extension [7] also perform an adaptive evaluation of test examples but focus on skipping units rather than layers. Feedback Networks [41] heavily shares parameters and allows early predictions in a recurrent process. However, their methods are less efficient in sharing computation. Our method is most inspired by MSDNet. Different from MSDNet, our proposed method focus on the difficulty diversity of image itself. And we also explore the frequency conflict occurred in a single model and therefore propose an original rolling strategy to handle the conflict.

III. METHODOLOGY

Problem. We investigate the failure cases which lead to poor performance in image SR. Given a 7-layers CNN, which
Fig. 2: Visualization results of a regular CNN for image SR. Left examples bring enormous PSNR(dB) promotion towards Bicubic. Right samples have higher absolute PSNR(dB) value as they contribute slight PSNR(dB) gain compared with Bicubic. It reveals that CNN is feasible to restore abundant texture images and bring great promotion, as patches with mild texture own a tiny upper bound.

Fig. 3: The features of bird and butterfly are visualized in (a) and (b). For better visualization, we reduce dimension to 1 along channel with max operation.

Fig. 4: Gradient properties of 10,000 severe and mild images. (a) Distribution of vertical axis gradient from high PSNR(dB) gain samples (e.g., severe images) in Figure.2a. (b) Distribution of vertical axis gradient from small PSNR(dB) gain cases (e.g., mild images) in Figure. 2b. We can easily distinguish successful and failure samples according to the gradient value.

SR task with gradient prior knowledge.

Overview of PRN. The proposed PRN aims at learning a framework, which can super-resolve images more efficiently. More specific, the proposed framework first label patches according to gradient prior. Thus, we can fetch the different patches from different feature level. Since the bottom convolutional stage has tiny receptive field and the mild patches is different from severe samples in term of frequency, we then relieve these problems by adopting a novel strategy to roll convolutional filters. Next, we first describe the definition of gradient prior and then present the setting of the proposed framework.

A. Gradient Prior

The proposed gradient prior is based on the observations that the failure samples in image SR usually have uniform gradient without sharp edges. As the samples with a uniform gradient contain rare pattern information and the upper bound for restoration is also pretty low, a simple and fast convolutional neural network can handle them well. We show the
vertical gradient distribution of 10,000 successful and failure samples in Figure.4a and 4b, respectively. It is obvious that mild samples have denser distribution among lower vertical gradient. And the distribution of severe images mainly lies on large value. With this gradient property, severe and mild samples can be distinguished. For an image, we describe the gradient property as follow:

\[ P(x) = \|G_{\text{ver}}(x)\|, \]

where \( x \) is the input image, \( G_{\text{ver}} \) counts the gradient along the vertical axis. \( P \) are the gradient prior knowledge, which also serves as a tag in our model. With the \( P \), PRN is able to separate a set of images into mild, moderate and severe patches.

\[
\begin{align*}
\text{Mild} & : P \leq \gamma_{\text{upper}} \\
\text{Moderate} & : \gamma_{\text{upper}} \leq P \leq \gamma_{\text{low}} \\
\text{Severe} & : \gamma_{\text{low}} \leq P.
\end{align*}
\]

The \( \gamma_{\text{upper}} \) and \( \gamma_{\text{low}} \) means the upper and low gradient threshold of \( P \) for separating the images. Moreover, we make an ablation study on the gradient threshold in section IV-B. Although \( P(x) \) is proposed based on the assumption that mild texture image is too simple to bring enormous gain, we show this prior can also be applied to accelerate image SR.

C. Rolling the convolutional filters

Although we can effectively enhance the image with the aforementioned model, we still find the following problems: 1) The receptive field of the early stage is tiny. When we try to improve the performance, the tiny receptive field of early stage become a bottleneck. 2) Frequency conflicts. The frequency domain of mild and moderate examples are significantly different from severe patches. When we train mild samples in the early stage of the network, the output of high level is influenced. Thus, we make an attempt to resolve the above questions by developing a novel rolling strategy.

Let \( \Theta \) be parameters of a CNN, it consists of four parts of parameters \( \{\Theta_l, \Theta_m, \Theta_s, \Theta_{up}\} \). Meanwhile, \( \Theta_l \) represents the early stage and consists of a convolutional layer. \( \Theta_m \) means the middle stage and has two layers. \( \Theta_s \) indicates the last stage and contains two convolutional layers and \( \Theta_{up} \) is the parameters of a deconvolutional layer. In addition, we define auxiliary two set of the parameter \( \Theta^{l}_{D} \) and \( \Theta^{l}_{D} \). More specific, \( \Theta^{l}_{D} \) represent a dilated convolution layer [40] with size of 64 × 5 × 5 and 1 dilation and \( \Theta^{l}_{D} \) means two dilated convolution layers with size of 64 × 3 × 3 and 1 dilation as well. We use \( P_{HR \downarrow} \) and \( P_{LR \downarrow} \) represent high-res and low-res patch, respectively. Three superscripts \( l \), \( m \) and \( * \) are used to distinguish the mild, moderate and severe samples respectively. For instance, \( P^{l}_{LR \downarrow} \) and \( P^{m}_{LR \downarrow} \) indicate the low-res patch annotated with mild and
severe tag. In sum, the enhancement toward severe patch can be defined as:

$$P_{out} = f(P_{LR}^s|\Theta_l, \Theta_m, \Theta_s, \Theta_{up})$$

where $f$ means the enhancement operation. As sketched in Fig. 6, the network will roll $\Theta_l$ and $\Theta_m$ with $\Theta_l^D$ and $\Theta_m^D$ and fetch the patch from different stage according to tag. More specific, suppose we input $P_{LR}^l$ into the network, the model will enhance the patch with $\Theta_l^D$ explicit. By that analogy, the enhancement process toward $P_{LR}^l$ and $P_{LR}^m$ can be formulated as:

$$P_{out} = f(P_{LR}^l|\Theta_l^D, \Theta_{up})$$

and

$$P_{out} = f(P_{LR}^m|\Theta_m^D, \Theta_{up})$$

With such flexible and content-adaptive rolling strategy, we not only resolve frequency conflicts but also increase the receptive field of early stage.

**D. End-to-end framework**

In contrast to training models with different datasets, the proposed model not only be able to fetch images from the different stage but also optimize each stage with specific prior. The whole procedure can be formulated as an end-to-end framework to accelerate speed. We have sketch detailed algorithm in Algorithm 1. Since the down-scaled mild sample is similar to its ground-truth, the model is unable to learn how to recover realistic details and textures w.r.t mild training examples. To resolve this question, we adopt the mild and moderate samples as the training pairs for $\Theta_l^D$ and $\Theta_m^D$. With such an efficient strategy, our model not only greatly improve the performance but also accelerate the training and testing speed.

**IV. EXPERIMENTS**

**Datasets.** To make full use of the parameters in PRN, we use VOC2012 [6] to pre-train our model. VOC2012 [6] contains 17,125 clear images, which are taken from natural scene. Then, we finetune our model with BSD200 [1], which contains 200 images and is close to the real-world scene. BSD200 [1] is augmented with scaling and rotation. We employ Set5, Set14, BSDS100, and Urban100 to evaluate our model.

**Implementation Details.** We use Xavier [8] initialize the parameters of the proposed model. Besides, the deconvolution layer is initialized according to the weight of Bicubic interpolation. We add pad with zero in each convolutional layer to assure the input tensor shares same size with the output. We convert all images from RGB to YCbCr and extract the Y channel for training. The training and testing images are cropped into $54 \times 54$ patches and down-scaled with the corresponding factor to obtain the input. For $54 \times 54$ patch, the $\gamma_{upper}$ and $\gamma_{low}$ are set as $1 \times 10^3$ and $3 \times 10^3$, respectively. In training, we set the batch size as 64 and learning rate is $1 \times 10^{-4}$ for all layers. In testing, we set the batch size as 1. The learning rate is reduced with factor $\times 10$ for every 300 epochs. We use leaky ReLU with a negative slope of 0.2 as the activate function. We perform our training and testing on a desktop computer with i7-4790 CPU, GTX980Ti GPU, and 32GB RAM.

**Multi-scale training.** Different from some state-of-the-arts [4], [5], [34], which conduct its model with single factor training, we adopt multi-scale learning strategy to train PRN. Specifically, multi-scaling learning is to train the
model with multiple down-sampling factors simultaneously. With the multi-scale learning, PRN can learn more contextual knowledge across different degeneration and achieves better performance.

A. Comparison with State-of-the-arts.

We compare our model with state-of-the-art methods, including A+ [38], SRF [32], SelFEx [13], RFL [33], SCN [39], SRCNN [5], LapSRN [19], VDSR [18], DRCN [17], and FSRCNN [4]. We adopt widely used quality metrics, e.g., PSNR and SSIM, to evaluate our model. For DRCN [17], we use our own implementation for comparison. For rest of other methods, we use their public code and model to obtain results.

As shown in table I, our model achieve superior performance among light-weight methods [4], [5], [13], [32], [33], [38], [39]. Compared with FSRCNN, our model achieve 0.13 dB and 0.19 dB promotion on BSDS100 with factor $2\times$ and $3\times$. Similarly, our model obtains 0.69 dB gain when compared with FSRCNN on Mange109 with factor $4\times$. With the limitation of the parameter, our model is weak than heavy inferences [17]–[19]. As sketched in figure I, our model shows slightly lower performance compared with huge model [17]–[19], but our speed is accelerated about several times. Therefore, the model is particularly competitive for mobile devices and applications.

We also show qualitative comparison in Figure. 8, 9, 10 and 11. For better visualization, we interpolate the chrominance space by bicubic to obtain color images. Compared with other methods, our approach can generate image clearer boundary and rich details.

B. Ablation study

In this section, we mainly investigate different settings of the proposed model and provide insights into the choice of hyper-parameters.

![Fig. 7: Efficiency and effectiveness analysis of different gradient threshold on BSDS100.](image)

**Gradient threshold.** We first analyze the setting of gradient threshold $\gamma_{upper}$ and $\gamma_{low}$ by investigating a wide range of potential values. In table II, we list all threshold we have compared. In fact, different gradient threshold may influence efficiency and effectiveness. In Fig. 7, we show the performance and efficiency of each setting. With the increment of $\gamma_{low}$, our model deal more moderate samples at an early stage, which accelerate speed but bring significant performance drop. A similar situation also occur when we increase the value of $\gamma_{upper}$. As the growth of $\gamma_{upper}$, the proposed model exhibit promising efficiency with degradation of performance. Since the middle or first stage is unable to deal severe samples well, we think too low $\gamma_{upper}$ and $\gamma_{low}$ may bring obvious performance drop. However, as illustrated in Fig. 7, the model becomes slower with a decrease of $\gamma$. To achieve a balance between efficiency and performance, we adopt ‘Our_YU2L2’ as default gradient threshold.

**Depth of different stage.** In this component, we compare the depth setting of each stage. In other words, we adjust the depth of $\Theta_1$ and $\Theta_m$ to verify our settings. In table III, we use different depth setting in the early and middle stage for comparison. As shown in table III, with the increase of $\Theta_1$, the model show sight PSNR promotion with slower efficiency. Since the early stage is adapted to handle mild patches only, we think too much parameter is meaningless for further promotion. In contrast, the middle stage $\Theta_m$ is utilized to deal with the moderate sample, which carries some texture and details. Therefore, the performance becomes worse when we reduce parameter of $\Theta_m$. Thus, we use a light-weight setting at an early stage and increase the parameters of the middle stage to exhibit an efficient framework.

**Rolling strategy.** In order to show the effectiveness of the proposed rolling strategy, we investigate models with and without rolling strategy. In table IV, ‘o Rolling’ means model without rolling strategy and ’ Rolling’ indicates the model with rolling component. Compared with ‘o Rolling’, the model with a rolling strategy achieve 0.09 dB improvement. Although our model has an auxiliary parameter, we can use them content-adaptively to assure efficiency. Thus, our model achieves superior performance and maintains competitive efficiency by adopting a rolling strategy.

**C. Limitations**

As our model achieves a good balance between effectiveness and efficiency, it still exists some limitations. To advance efficiency, we need to crop the image into smaller images and reconstruct them at last. Thus, our model needs additional time to accomplish the reconstruction procedure. The reconstruction cost is far less than the model computational cost, and we have count the reconstruction time into time complexity in efficiency analysis. Besides, the acceleration is influenced by datasets. For instance, our model can accelerate the speed greatly on BSDS100 or DIV2K as the images in BSDS100 or DIV2K have plenty of blank and mild region. Similar acceleration cannot occur in General-100 as the images in General-100 are full with texture and edges. However, we think the majority of nature images, which is closed to BSDS500 and DIV2K, are occupied with a certain percentage of the blank or mild region. Therefore, our model can perform similar acceleration in real-world scenarios.
TABLE I: The PSNR and SSIM results of different approaches on Set5, Set14, BSDS100 and Urban100 with down-sampling factor ×2, ×3 and ×4. We use the black to label the first place.

| Algorithm | Scale | Set5 PSNR | SSIM | Set14 PSNR | SSIM | BSDS100 PSNR | SSIM | URBAN100 PSNR | SSIM |
|-----------|-------|-----------|------|------------|------|--------------|------|---------------|------|
| Bicubic   | 2x    | 33.69     | 0.931| 30.25      | 0.870| 29.37        | 0.844| 26.89         | 0.841|
| A+        | 2x    | 36.60     | 0.955| 32.32      | 0.906| 31.24        | 0.887| 29.25         | 0.895|
| RFL       | 2x    | 36.59     | 0.954| 32.29      | 0.905| 31.18        | 0.885| 29.14         | 0.891|
| SelfEx    | 2x    | 36.60     | 0.955| 32.24      | 0.904| 31.20        | 0.887| 29.55         | 0.898|
| SRCNN     | 2x    | 36.72     | 0.955| 32.51      | 0.908| 31.38        | 0.889| 29.53         | 0.896|
| SCN       | 2x    | 36.58     | 0.954| 32.35      | 0.905| 31.26        | 0.885| 29.52         | 0.897|
| FSRCNN    | 2x    | 37.05     | 0.956| 32.66      | 0.909| 31.53        | 0.892| 29.88         | 0.902|
| Our       | 2x    | 37.09     | 0.957| 32.90      | 0.910| 31.66        | 0.893| 30.23         | 0.909|
| Bicubic   | 3x    | 30.41     | 0.869| 27.55      | 0.775| 27.22        | 0.741| 24.47         | 0.737|
| A+        | 3x    | 32.62     | 0.909| 29.15      | 0.820| 28.31        | 0.785| 26.05         | 0.799|
| RFL       | 3x    | 32.47     | 0.906| 29.07      | 0.818| 28.23        | 0.782| 25.88         | 0.792|
| SelfEx    | 3x    | 32.66     | 0.910| 29.18      | 0.821| 28.30        | 0.786| 26.45         | 0.810|
| SRCNN     | 3x    | 32.78     | 0.909| 29.32      | 0.823| 28.42        | 0.788| 26.25         | 0.801|
| SCN       | 3x    | 32.62     | 0.908| 29.16      | 0.818| 28.33        | 0.783| 26.21         | 0.801|
| FSRCNN    | 3x    | 33.18     | 0.914| 29.37      | 0.824| 28.53        | 0.791| 26.43         | 0.808|
| Our       | 3x    | 33.52     | 0.916| 29.64      | 0.828| 28.72        | 0.794| 26.75         | 0.815|
| Bicubic   | 4x    | 28.43     | 0.811| 26.01      | 0.704| 25.97        | 0.670| 23.15         | 0.660|
| A+        | 4x    | 30.32     | 0.860| 27.34      | 0.751| 26.83        | 0.711| 24.34         | 0.721|
| RFL       | 4x    | 30.17     | 0.855| 27.24      | 0.747| 26.76        | 0.708| 24.20         | 0.712|
| SelfEx    | 4x    | 30.34     | 0.862| 27.41      | 0.753| 26.84        | 0.713| 24.83         | 0.740|
| SRCNN     | 4x    | 30.50     | 0.863| 27.52      | 0.753| 26.91        | 0.712| 24.53         | 0.725|
| SCN       | 4x    | 30.41     | 0.863| 27.39      | 0.751| 26.88        | 0.711| 24.52         | 0.726|
| FSRCNN    | 4x    | 30.72     | 0.866| 27.61      | 0.755| 26.98        | 0.715| 24.62         | 0.729|
| Our       | 4x    | 31.08     | 0.875| 27.89      | 0.762| 27.17        | 0.728| 24.86         | 0.733|

Fig. 8: Qualitative comparison on 'butterfly' with the scaling factor of 4. We use red and blue to label best two results, respectively. Best viewed by zooming in the electronic version.
Fig. 9: Qualitative comparison on 'bird' with the scaling factor of 4. We use red and blue to label best two results, respectively. Best viewed by zooming in the electronic version.

Fig. 10: Qualitative comparison on 'baboon' with the scaling factor of 3. We use red and blue to label best two results, respectively. Best viewed by zooming in the electronic version.
TABLE II: We have compared a wide range of potential gradient threshold. Meanwhile, L indicates $\gamma_{\text{low}}$ and U is $\gamma_{\text{upper}}$. The suffix number along L and U means different threshold value.

| $\gamma$ | L1 | L2 | L3 | L4 | U1 | U2 | U3 | U4 |
|----------|----|----|----|----|----|----|----|----|
| Value(10$^x$) | 1  | 2  | 5  | 7  | 3  | 5  | 8  | 10 |

TABLE III: Comparison of different depth toward early and middle stage on BSDS100. The superscripts of $\Theta$ mean different depth of each stage. The subscripts of $\Theta$ indicate different stage.

| $\Theta_1^i$ | $\Theta_2^i$ | $\Theta_3^i$ | $\Theta_4^i$ | $\Theta_5^m$ | $\Theta_6^m$ |
|--------------|--------------|--------------|--------------|--------------|--------------|
| PSNR         | 27.12        | 27.13        | 27.15        | 27.05        | 27.12        | 27.15        |
| Time         | 1.81         | 1.91         | 2.21         | 1.48         | 1.81         | 1.99         |

TABLE IV: Comparison of with and without rolling strategy on BSDS100.

| Rolling | o Rolling |
|---------|-----------|
| PSNR    | 27.03     | 27.12     |

V. CONCLUSION AND FURTHER WORK

In this article, to address efficiency problem in image SR, we have proposed an end-to-end gradient-aware rolling network. Our model mainly incorporates gradient prior to the image itself and content-adaptively utilize each stage of the deep neural network to super-resolve corrupted images. Moreover, we have proposed a rolling strategy, which super-resolve images with the different set of filters, to resolve frequency conflicts problem. Experiments have shown that our framework not only obtains competitive performance but also achieve appealing efficiency.

There are several directions for us to extend our work. First, we can introduce adversarial loss or perceptual loss in each stage, aiming to restore more realistic details and texture. Second, considering exist framework have to crop the image into patches, we intend to propose a more general framework, which can content-adaptively process different region with the different stride of convolution operation, to boost efficiency.

REFERENCES

[1] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 33(5):898–916, May 2011.

[2] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for fast test-time prediction. CoRR, abs/1702.07811, 2017.

[3] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. Computer Science, 2014.

[4] Chao Dong, Change Loy Chen, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. pages 391–407, 2016.

[5] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In ECCV, pages 184–199, 2014.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results.

[7] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry P Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In CVPR, volume 2, page 7, 2017.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256, 2010.
Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Jia Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In Computer Vision and Pattern Recognition, pages 5197–5206, 2015.

Ilay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In Advances in neural information processing systems, pages 4107–4115, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. 2017.

Gao Huang, Zhuang Liu, Van Der Maaten Laurens, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. 2017.

Gao Huang, Zhuang Liu, Van Der Maaten Laurens, and Kilian Q Weinberger. Densely connected convolutional networks. pages 2261–2269, 2016.

Jia Bin Huang, Abhishekg Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In Computer Vision and Pattern Recognition, pages 5197–5206, 2015.

Yoshua Bengio. Binarized neural networks. In Advances in neural information processing systems, pages 4107–4115, 2016.

Wenwu Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. pages 1874–1883, 2016.

Wenwu Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1874–1883, 2016.

Ying Tai, J Ian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network.

Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. pages 4549–4557, 2017.

Radu Timofte, Vincent De, and Luc Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In IEEE International Conference on Computer Vision, pages 1920–1927, 2013.

Zhaowen Wang, Ping Liu, Jianchao Yang, Wei Han, and Thomas Huang. Deep networks for image super-resolution with sparse prior. pages 370–378, 2015.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. 2016.

Amir R Zamir, Te-Lin Wu, Lin Sun, William B Shen, Bertram E Shi, Jitendra Malik, and Silvio Sarevares. Feedback networks. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 1808–1817. IEEE, 2017.

Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration.

Ting Zhang, Guo Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions for deep neural networks. 2017.

Xiaoyang Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. 2017.

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. 2018.