Progressive Splitting and Upscaling Structure for Super-Resolution

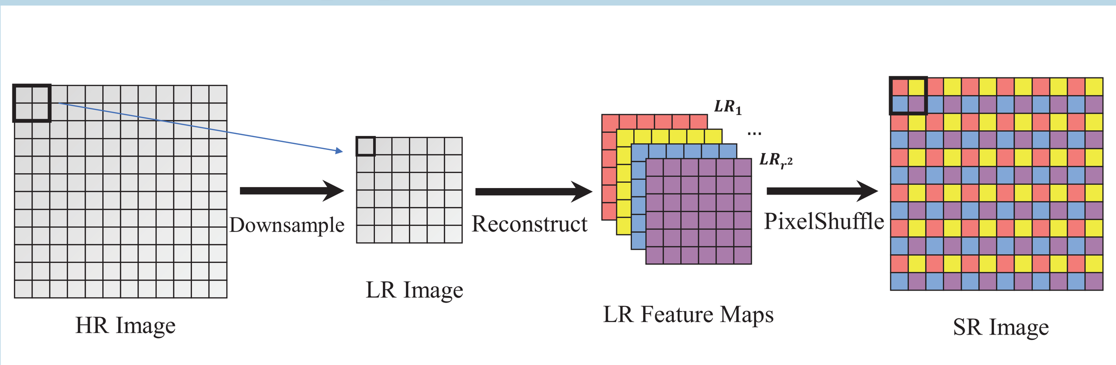
Qiang Li, Tao Dai, Shu-Tao Xia
liqiang19@mails.tsinghua.edu.cn, daitao.edu@gmail.com, xiast@sz.tsinghua.edu.cn

Problem

Most super-resolution (SR) methods focus on the design of network architecture and adopt a sub-pixel convolution layer at the end of network, but few have paid attention to exploring potential representation ability of upscaling layer.

Method

Sub-pixel convolution layer aggregates several low resolution (LR) feature maps and builds super-resolution (SR) images in a single step. However, those LR feature maps share similar patterns as they are extracted from a single trunk network. In this paper, we propose a progressive splitting and upscaling structure (PSUS) for SR task. It works with a certain basic block and aims at generating decoupled SR features progressively. It uses fewer parameters and lower computational cost, whose details are shown in the paper.



Overview

Flexible Structure We propose a progressive splitting and upscaling structure (PSUS) for image SR to explore the potential representation ability of upscaling layer.

Novel Splitting Strategy We propose a progressive splitting module (PSM) which can produce decoupled deep features using approximately the same computational cost.

Efficient Upscale Module We propose a multipath upscale module (MUM) which aggregates LR features. Besides, we propose a transition strategy to further reduce computational cost and parameters.

References

[1] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, *Enhanced deep residual networks for single image super-resolution*, Proceedings of the IEEE conference on computer vision and pattern recognition workshops, 2017, pp. 136–144.

[2] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu: *Image super-resolution using very deep residual channel attention networks*, Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 286–301.

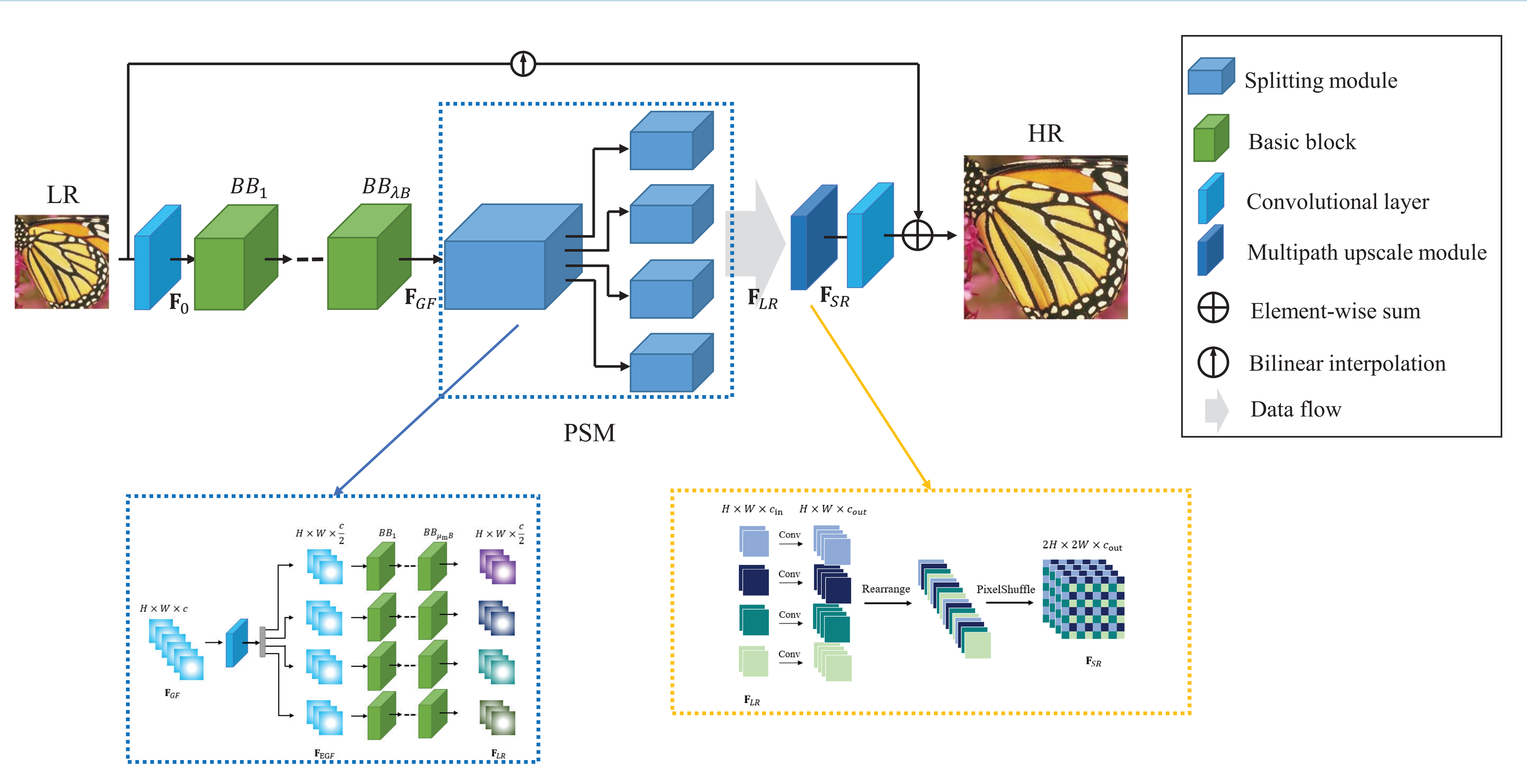
[3] A. Shocher, N. Cohen, and M. Irani: *"zero-shot" super-resolution using deep internal learning*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 3118–3126.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China under Grant 61771273, Guangdong Basic and Applied Basic Research Foundation on 2019A1515110344, the China Postdoctoral Science Foundation under Grant 2019M660645, the R&D Program of Shenzhen under Grant JCYJ20180508152204044, and the project "PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019)".

Proposed Architecture

Our proposed PSUS consists of four parts: shallow feature extraction, global feature extraction, progressive splitting module (PSM) and multipath upscale module (MUM). We use $\mathbf{I}_{LR}$ and $\mathbf{I}_{SR}$ to denote input and output of network. One convolutional layer is used to extract the shallow feature $\mathbf{F}_0$ from LR input image. Assuming that corresponding chain-like model stacks B basic blocks, we use λ ($0 < \lambda < 1$) of them to extract global feature. Further, we adopt a PSM to progressively decouple features and generate $\mathbf{F}_{LR_1}, \mathbf{F}_{LR_2} \cdots \mathbf{F}_{LR_{r^2}}$. Then, we have got r^2 groups of features which correspond to each position of $r \times r$ patch respectively. What MUM does is to aggregate these features and generate C SR features. The architecture of PSM and MUM have been shown in the figure. More details could been found in the paper.



Experiments

we compare the performance of our method to three widely-used models: a small model EDSR-baseline from [1], a large model RCAN[2] and an unsupervised SR model ZSSR[3].

We conduct experiments on a small model EDSR-baseline firstly. It is a single-scale model and only contains 16 ResBlocks. For PSUS with ResBlock, we study the effects of different λ for $\times 2$ model and then choose proper values of μ_1, μ_2 for $\times 4$ model. Quantitative metrics are presented. λ denotes the ratio of basic blocks using for extracting global feature. For $\times 2$ model, its network architecture is determined by the single hyperparameter λ. We retrain EDSR-baseline model in our environment for fair comparison and train $\times 2$ PSUS with $\lambda \in \{0.875, 0.75, 0.5\}$. PSNR and SSIM are shown in Table I. As for $\times 4$, results are shown in Table III and Table IV.

TABLE I: Quantitative results (scale $\times 2$) of our PSUS with different λ and baseline. PSNR(dB) and SSIM are tested on Y channel without self-ensemble [9]. DIV2Kval denotes DIV2K validation set. Best results are **highlighted**.

	Baseline PSNR/SSIM	$\lambda = 0.875$ PSNR/SSIM	$\lambda = 0.75$ PSNR/SSIM	$\lambda = 0.5$ PSNR/SSIM
Set5	37.96/0.9604	37.96/0.9603	**37.98**/0.9603	**37.98/0.9604**
Set14	33.51/0.9168	33.48/0.9163	33.52/**0.9173**	**33.53**/0.9172
BSD100	32.13/0.8991	32.12/0.8989	**32.15/0.8994**	**32.15**/0.8993
Urban100	31.80/0.9255	31.86/0.9261	**31.96**/0.9268	31.95/**0.9269**
DIV2Kval	36.04/0.9449	36.06/0.9450	36.10/0.9453	**36.11/0.9454**
Average	34.29/0.9294	34.30/0.9293	**34.34/0.9298**	**34.34/0.9298**

TABLE III: Quantitative metrics of model complexity and computational cost for different $\times 4$ models.

	EDSR-Baseline	PSUS with ResBlock
Params	1.518M	1.483M (-2.3%)
FLOPs	257.47G	224.27G (-12.9%)

TABLE IV: PSNR(dB) and SSIM results (scale $\times 4$) of baseline and our proposed PSUS. Best results are **highlighted**.

	Baseline (from scratch) PSNR/SSIM	Baseline (from pre-trained x2) PSNR/SSIM	PSUS with ResBlock PSNR/SSIM
Set5	32.09/0.8936	32.11/0.8937	**32.13/0.8938**
Set14	28.53/0.7807	**28.56/0.7816**	28.50/0.7805
BSD100	**27.55**/0.7352	27.54/**0.7357**	**27.55**/0.7354
Urban100	25.95/0.7817	26.00/**0.7839**	**26.01/0.7839**
DIV2Kval	30.38/0.8366	30.40/0.8373	**30.42/0.8375**

We then conduct experiments on a state-of-the-art model RCAN to see whether our method can obtain similar improvement on large models. Quantitative metrics and visual comparison will be presented. As shown in Figure 5, during the first 2×10^5 iterations, our method can converge much faster than baseline. Some visual Results are shown in Figure 7. More results could been found in the paper.

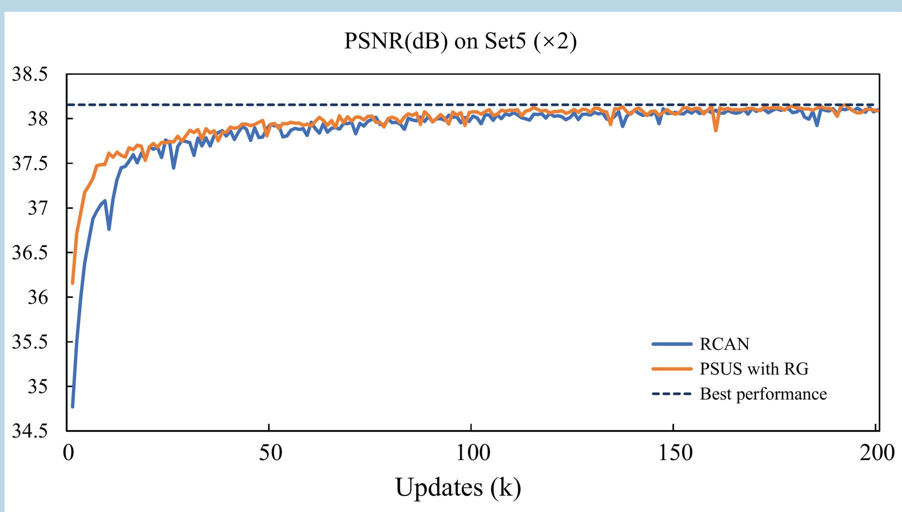


Fig. 5: PSNR on validation set of $\times 2$ models during first 2×10^5 iterations of training.

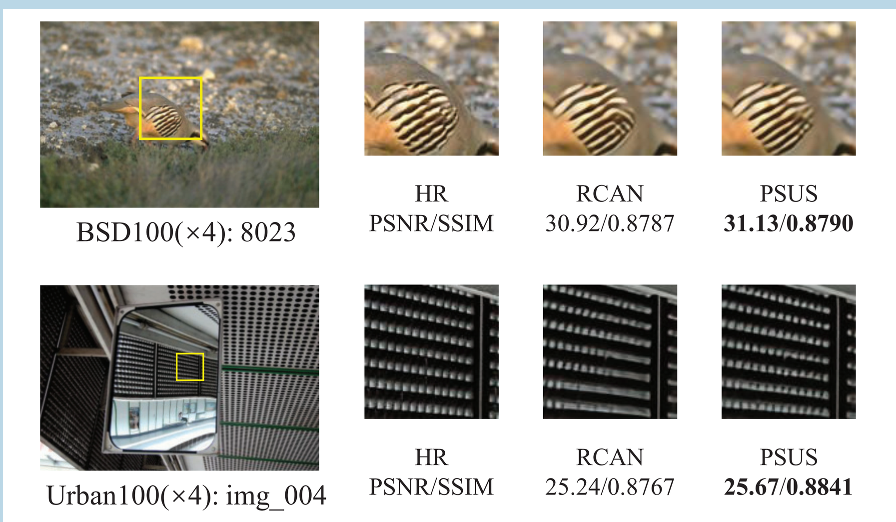


Fig. 7: Visual comparison for $\times 4$ SR. Best results are **highlighted**.