Appendix for Accelerating Image Super-Resolution Networks with Pixel-Level Classification

Jinho Jeong[1], Jinwoo Kim[1], Younghyun Jo[2], and Seon Joo Kim[1]

[1] Yonsei University
[2] Samsung Advanced Institute of Technology

1 Adaptive Decision Making (ADM)

During the inference phase of our PCSR, we provide additional functionality: Adaptive Decision Making (ADM), which automatically assigns pixels to proper-sized branches. While a simple approach is to allocate the pixel to the branch with the highest probability, ADM differs by taking into account statistical values of probabilities across the entire image. The value for each i-th pixel in the image initially determined through $sum_{0\leq j<\lfloor (M+1)/2 \rfloor} p_{i,j}$ to represent the restoration difficulty of that pixel, considering $U_{j\in[0,\lfloor (M+1)/2 \rfloor)}$ as heavy upsamplers. Subsequently, these difficulty values are used to perform k-means clustering with M clusters and each clusters are assigned to the corresponding branch.

We show the potential of ADM through Fig. 1. While the simple approach fixes the threshold at 0.5 regardless of images, ADM adaptively forms the threshold at the point where the density of difficulty starts to sufficiently decrease by clustering areas with high value density. That is, ADM avoids regions where even minor variations in the threshold could lead to sensitive changes in pixel allocation. It instead allows the threshold to be established in a section that remains stable against these variations, ensuring a more consistent allocation. Additionally, since only a few iterations (about 2-7 iters per image) are required for clustering to converge, the additional overhead by ADM is negligible.

2 More Experiments

2.1 Results on other benchmarks

We provide results for other benchmarks including Set14, B100, and Manga109. As shown in Tab. 1, our method is still efficient even for images of moderate size, compared to patch-based methods.

2.2 Running Time Comparison

Tab. 2 compares the running time between the patch-based methods and our method. Although the running time of ours is much faster, note that all methods primarily aim to reduce FLOPs, and the implementations are not fully optimized for the running time. We will look into more efficient implementation.

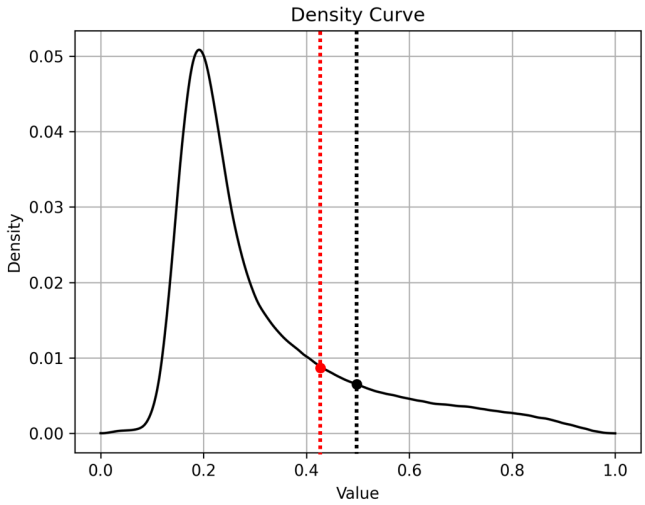

Fig. 1: Difficulty density curve for the image "0855" (DIV2K) with M=2 on $\times$4. The range of values are divided into 100 bins, with density calculated as the count of values per bin divided by the total value count. The density, associated with each bin's center, is interpolated to form a smooth curve. Each dotted line indicates threshold for assigning pixels: pixels left of a line go to the light upsampler, those to the right to the heavy upsampler. The black dotted line represents a threshold (= 0.5) of simple approach (*i.e.*, allocating pixels to the upsampler with the highest probability), while red dotted line indicates an adaptively determined threshold by ADM.

3 More Ablation Studies

3.1 Impact of the condition for pixel-wise refinement

Pixel-wise refinement is designed to minimize artifacts by adjusting the RGB values of pixels assigned to light upsamplers to the average RGB value of their neighbors if any adjacent pixels are assigned to heavy upsamplers. We investigate how many neighboring pixels should be allocated to heavy upsamplers to effectively reduce artifacts while maintaining performance, as shown in Tab. 3.

Interestingly, we observe negligible performance degradation for any condition, even when all the pixels assigned to light upsamplers are replaced regardless of the status of neighboring pixels (*i.e.*, #h=0). According to the table, while there is a slight decrease in performance when at least one neighboring pixel is allocated to heavy upsamplers (*i.e.*, #h=1), this condition results in a greater number of replaced pixels, which is beneficial for artifact removal. Therefore, we choose #h=1 and always activate refinement in our evaluation.

3.2 Impact of the LIIF Upsampler

We compare between LIIF-based and CNN (or pixelshuffle)-based upsamplers in Tab. 4. The performance of the model can be higher with the LIIF upsampler

Table 1: PSNR and FLOPs for additional benchmarks on ×4.

Model	Set14(dB)	FLOPs	B100(dB)	FLOPs	Manga109(dB)	FLOPs
CARN	26.52	13.53G	26.37	7.86G	27.93	64.01G
+ClassSR	26.48	13.25G	26.32	6.88G	27.86	68.04G
+ARM	26.48	14.12G	26.32	6.56G	27.88	69.79G
+PCSR	26.48	**8.14G**	26.32	**4.19G**	27.86	**40.66G**

Table 2: Comparison of running time per image on ×4, when the performance of ARM and PCSR is set to be the same as ClassSR.

Model(CARN)	Urban100	Test2K	Test4K
+ClassSR	1994ms	4595ms	19072ms
+ARM	518ms	1069ms	4608ms
+PCSR	**45ms**	**62ms**	**203ms**

than the original (*e.g.*, FSRCNN, CARN), but for SRResNet, the performance is same or even lower than the original. Hence, we argue that the adoption of the LIIF does not guarantee the higher performance.

4 Effectiveness on the Recent Lightweight Model

To further demonstrate PCSR's broad applicability and efficiency, we apply our PCSR method to the recent lightweight model, BSRN [3]. BSRN is the model that won first place in the model complexity track of the NTIRE 2022 Efficient SR Challenge, utilizing separable convolutions to enhance its scalability. The result is shown in Tab. 5, illustrating that PCSR achieves performance comparable on several large image-based benchmarks while using fewer FLOPs. This highlights the versatility and effectiveness of our approach.

5 More Visual Comparisons

In this section, we provide additional visual comparisons to ClassSR and ARM, along with PSNR values and FLOPs, demonstrating our method's efficiency and capability. In Fig. 2b, the patch-based methods engage in over-computation, which results in unnecessary computational expense. Our method saves computations by efficiently allocating resources on a pixel basis while maintaining high quality. In Fig. 2c, while under-computation by patch-based methods results in blurry outcomes, our method differentiates difficulties with precision, producing sharper and more defined restorations. For Fig. 2d and 3d, instead of applying moderate computation uniformly across patches, our method focuses on challenging areas, achieving higher image quality with comparable computational cost. Across various cases, the patch-based methods struggle with mixed restoration difficulties within a patch, but our pixel-level classification manages these variations effectively, improving both PSNR and FLOPs efficiency.

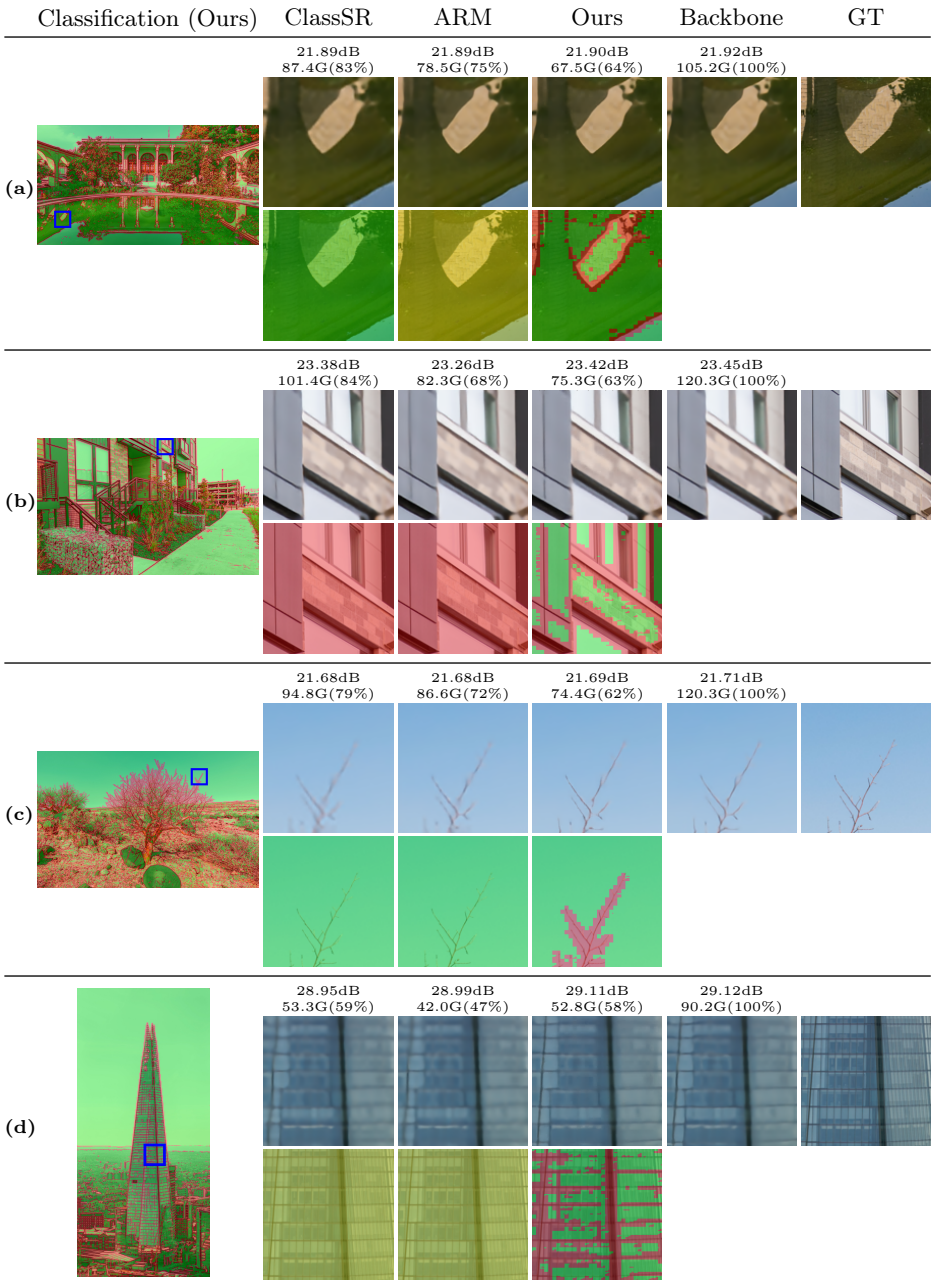

Fig. 2: Qualitative results of the previous methods [1, 2] and our method with $\times 4$ SR on Test2K.

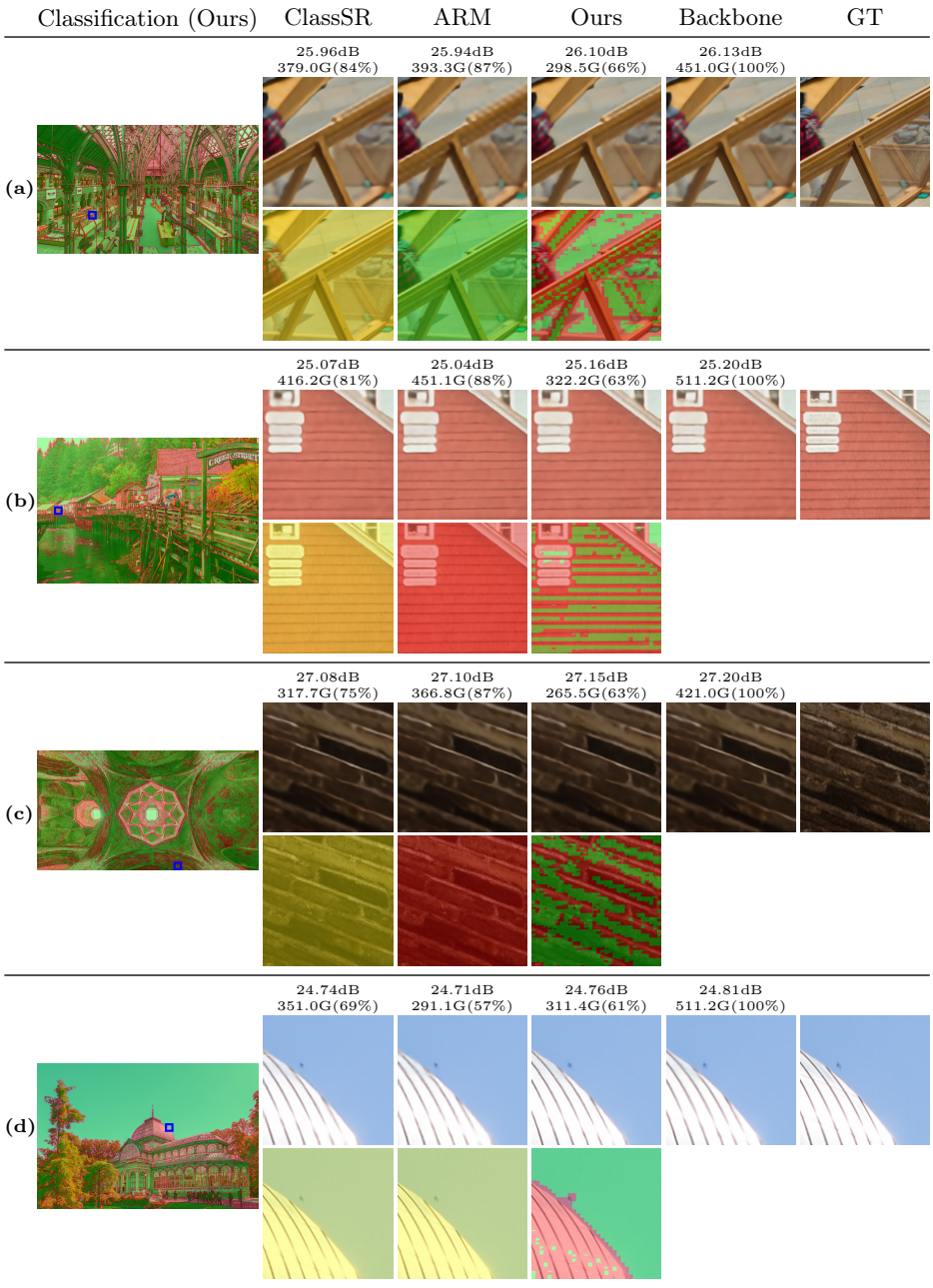

Fig. 3: Qualitative results of the previous methods [1, 2] and our method with $\times 4$ SR on Test4K.

Table 3: Variation in PCSR performance on Test2K (×4) depending on the condition for pixel-wise refinement. Here, "#h" denotes the threshold number of neighboring pixels allocated to heavy upsamplers required around a pixel to trigger its replacement. #h=9 can be considered as the performance where no refinement is performed.

Model	CARN-PCSR					
#h	0	2	4	6	8	9
PSNR (dB)	25.995	26.011	26.016	26.021	26.022	26.022

Table 4: Comparison between pixel-shuffle upsampler and LIIF upsampler on ×4. **MAX** denotes maximum PSNR and FLOPs by our method.

Model	Test2K(dB)	FLOPs	Test4K(dB)	FLOPs	Urban100(dB)	FLOPs
FSRCNN	25.69	45.3G	26.99	185.3G	23.05	19.9G
+PCSR(MAX)	25.69	44.5G	27.01	181.8G	23.27	19.6G
CARN	26.03	112.0G	27.45	457.8G	24.03	49.3G
+PCSR(MAX)	26.05	114.4G	27.47	467.7G	24.09	50.3G
SRResNet	26.24	502.9G	27.71	2056.2G	24.65	221.3G
+PCSR(MAX)	26.24	507.9G	27.71	2076.6G	24.63	223.5G

Table 5: Comparison of BSRN with and without PCSR on scale ×4 SR.

Models	Params.	Test2K(dB)	GFLOPs	Test4K(dB)	GFLOPs	Urban100(dB)	GFLOPs
BSRN	352K	26.16	66.0 (100%)	27.52	270.0 (100%)	24.43	29.1 (100%)
BSRN-PCSR	198K	26.10	51.8 (78%)	27.52	208.4 (77%)	24.29	23.6 (81%)

References

1. Chen, B., Lin, M., Sheng, K., Zhang, M., Chen, P., Li, K., Cao, L., Ji, R.: Arm: Any-time super-resolution method. In: European Conference on Computer Vision. pp. 254–270. Springer (2022)
2. Kong, X., Zhao, H., Qiao, Y., Dong, C.: Classsr: A general framework to accelerate super-resolution networks by data characteristic. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12016–12025 (2021)
3. Li, Z., Liu, Y., Chen, X., Cai, H., Gu, J., Qiao, Y., Dong, C.: Blueprint separable residual network for efficient image super-resolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 833–843 (2022)