LSR: A LIGHT-WEIGHT SUPER-RESOLUTION METHOD

Wei Wang ¹, Xuejing Lei ¹, Yueru Chen ², Ming-Sui Lee ³, C.-C. Jay Kuo ¹

University of Southern California, Los Angeles, California, USA¹ Peng Cheng Laboratory, Shenzhen, Guangdong, China² National Taiwan University, Taipei, Taiwan³

ABSTRACT

A light-weight super-resolution (LSR) method from a single image targeting mobile applications is proposed in this work. LSR predicts the residual image between the interpolated low-resolution (ILR) and high-resolution (HR) images using a self-supervised framework. To lower the computational complexity, LSR does not adopt the end-to-end optimization deep networks. It consists of three modules: 1) generation of a pool of rich and diversified representations in the neighborhood of a target pixel via unsupervised learning, 2) selecting a subset from the representation pool that is most relevant to the underlying super-resolution task automatically via supervised learning, 3) predicting the residual of the target pixel via regression. LSR has low computational complexity and reasonable model size so that it can be implemented on mobile/edge platforms conveniently. Besides, it offers better visual quality than classical exemplar-based methods in terms of PSNR/SSIM measures.

Index Terms— Super-resolution, Mobile Computing, Green Learning

1. INTRODUCTION

Single image super-resolution (SISR) [1] is an intensively studied topic in image processing. It aims at recovering a high-resolution (HR) image from its low-resolution (LR) counterpart. SISR finds wide real-world applications such as remote sensing, medical imaging, and biometric identification. Besides, it attracts attention due to its connection with other tasks (e.g., image registration, compression, and synthesis).

SISR is an ill-posed problem since multiple HR patches can map to the same LR patch. To solve this one-to-many mapping problem, SISR is typically formulated as a regularized optimization problem or a generative problem with supervised learning. For the former, one may impose priors to regularize the ill-posed problem, yet the performance improvement is limited. For the latter, there are two main approaches: exemplar-based (or dictionary-based) methods and deep-learning (DL) methods.

DL-based super-resolution methods have been dominating in the field since 2015 [2]. They have been intensively studied in the last eight years. They offer better HR images in terms of PSNR/SSIM quality metrics at the cost of higher network parameters and larger computational complexity. One of the main applications of the SR

techniques is mobile platforms and consumer electronics (e.g., smart TVs). DL-based SR solutions cannot be easily implemented on resource-constrained computational platforms due to the price consideration.

To address this problem, a light-weight super-resolution (LSR) method is proposed in this work. LSR predicts the residual image between the interpolated low-resolution (ILR) and HR images using a self-supervised learning paradigm. LSR does not adopt the end-to-end optimization deep networks. Instead, it consists of three cascaded modules. First, it creates a pool of rich and diversified representations in the neighborhood of a target pixel via unsupervised learning. Second, it selects a subset from the representation pool that is most relevant to the underlying super-resolution task automatically via supervised learning. Third, it predicts the residual of the target pixel based on the selected features through regression via classical machine learning such as the XGBoost regressor. LSR offers visual quality that is better than exemplar-based methods and comparable with the entry-level DL-based SR solution, SRCNN, in terms of PSNR/SSIM measures.

It is worthwhile to highlight the value of this work. Our main contributions lie in the low computational complexity of the proposed LSR method. As presented in the experimental section, there are two versions of the LSR method, namely, LSR V1 and LSR V2. Both have around 380K parameters. We use the number of floating-point operations per pixel (FLOPs/pixel) to measure the complexity in inference. LSR V1 and LSR V2 demand 9.28K and 3.83K FLOPs/pixel, respectively. For complexity benchmarking, we choose two well-known and representative DL-based SR solutions; i.e., SRCNN and VDSR. SRCNN and VDSR demand 114K and 1.33M FLOPs/pixel, respectively. The LSR method has a clear advantage over DL-based solutions when being deployed on the mobile/edge platforms.

The rest of the paper is organized as follows. Related work is reviewed in Sec. 2. The proposed LSR method is presented in Sec. 3. Experimental results are shown in Sec. 4. Finally, concluding remarks and future research directions are discussed in Sec. 5.

2. REVIEW OF RELATED WORK

Exemplar-based Methods. Image patches are viewed as examples of local regions. Patches are partitioned and represented in the form of dictionary atoms. Finally, proper mappings from LR-to-HR patches are developed inside each partition. Examples include [3, 4, 5, 6, 7]. Sometimes, priors are leveraged for patch partitioning [8]. Since the dictionary size can be expanded flexibly, they are non-parametric methods. There are limitations with exemplar-based methods. First, the LR-to-HR mapping is based on hand-crafted features. There is no clear guideline in patch sizes for mapping learning.

The authors acknowledge the gift support from MediaTek Inc. as well as the Center for Advanced Research Computing (CARC) at the University of Southern California for providing computing resources that have contributed to the research results reported within this publication. URL: https://carc.usc.edu.

Second, the training of the LR-to-HR mapping is time-consuming [4, 5]. Last, the quality of their enhanced SR images is inferior to that achieved by modern DL-based methods.

DL-based Methods. The application of DL to the SISR problem can be traced back to SRCNN in 2015 [2]. Substantial advances have been made along this direction in the last eight years, e.g., [9, 10, 11, 12, 13, 14, 15, 16, 17]. In earlier years, the focus was on achieving higher performance (namely, better PSNR and SSIM) [2, 9, 10, 11, 12, 13]. Research on efficiency has been considered in recent years, e.g., [14, 15]. Other SISR-related problems have also been explored, e.g. unknown degradation kernels [16], magnification with a non-integer factor [17], etc. Although DL-based methods offer significant performance breakthrough, it is a major challenge to apply them to practical SR problems in the mobile/edge devices due to their heavy computational and memory costs. Besides, they lack mathematical transparency.

Green Learning. Green learning [18] is an emerging learning paradigm emphasizing lower computational complexities and smaller model sizes. It has a modular design that consists of three cascaded modules: 1) unsupervised representation learning, 2) supervised feature learning, and 3) supervised decision learning. For unsupervised representation learning, Kuo et al. interpreted the convolution operations in convolutional neural networks (CNNs) as joint spatial-spectral signal transforms in [19, 20, 21] and proposed two one-stage data-driven transforms, the Saak transform [20] and the Saab transform [21]. To achieve multi-stage signal transforms, Kuo developed a successive-subspace-learning (SSL) strategy in [20, 21]. For supervised feature learning, the problem of selecting the most discriminant (or relevant) features from the pool of rich representations for some classification (or regression) tasks based on user's labels is examined in [22]. Green learning has been successfully applied to many applications. Our LSR method follows the same pipeline as elaborated below.

3. PROPOSED LSR METHOD

For self-supervised SR, LR images are obtained from HR images via bicubic down-sampling in training and test image sets. Following the standard pipeline, we only focus on the luminance (or Y) component. As a pre-processing step, a Lanczos interpolation is applied to LR images to yield interpolated LR (ILR) images whose resolution is the same as HR images. To regularize the ill-posedness of the problem, LSR uses the neighborhood of a target pixel to predict its residue, which is the difference between HR and ILR images at the pixel. Thus, the input and output to the proposed LSR system are an ILR patch (of size 15×15) and the residual value of its center pixel, respectively.

It is desired to divide pixels into easy and hard two classes. Pixels in smooth regions are easy samples. Their residual values are small since the interpolation can predict their values quite well. Furthermore, their residuals can be predicted using a simple model of lower complexity. Pixels in complicated regions such as edges and textures are hard samples. Their residual values are larger, and a more complicated model is required. To exploit this property, we develop a simple mechanism to partition pixels based on the variance of their neighborhood. A pixel whose neighborhood has a smaller variance is an easy one. Otherwise, it is a hard one. We focus on hard samples for the rest of this section. The same idea applies to easy samples but the processing can be greatly simplified.

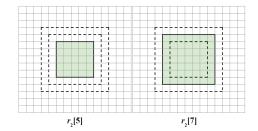


Fig. 1. Illustration of central Saab representations of Type 2, where $r_2[n]$ denotes Representation Type 2 with size $n \times n$.

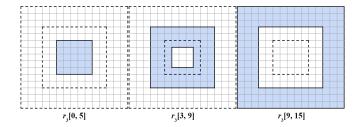


Fig. 2. Illustration of ring-wise Saab representations of Type 3, where $r_3[n_1, n_2]$, $n_2 > n_1$, denotes the difference between the square of width n_2 and the square of width n_1 .

3.1. Module 1: Unsupervised Representation Learning

The objective of the first module is to generate a rich and diversified set of representations. Five types of representations are collected with some justifications.

- Type 1: spatial representations. For a patch of size 15×15 , it has 225 pixels values as representations of Type 1.
- Type 2: central Saab representations. Since pixels closer to the central pixel are more important than distant pixels, we consider two windows of sizes 5×5 , 7×7 as shown in Fig.1 and apply the Saab transform to pixels within each window to yield the central Saab representations. A window of size $n \times n$ will yield n^2 Saab coefficients as the representations of Type 2^{-1} .
- Type 3: ring-wise Saab representations. Ring-shaped neighborhoods are introduced to complement the center-shaped neighborhoods as shown in Fig. 2. We apply one-stage Saab transforms with 3 × 3 blocks at stride 1 on r₃[0, 5] and at stride 3 on outter ring regions, leading to one DC coefficient and 8 AC coefficients per block. These 9-channel responses are decoupled due to PCA.
- Type 4: Haar filtering followed by channel-wise PCA representations. For a neighborhood of size 2 × 2, the Haar filterbank yields four responses and each response is treated as one channel. Then, PCA is applied to each channel. Type 4 representation is derived from the original Haar response and PCA coefficients.
- Type 5: Laws filtering followed by channel-wise PCA representations. For a neighborhood of 3 × 3, Laws' filterbank [23] yields nine responses and each response is treated as one

 $^{^1}$ A Saab transform of size $n \times n$ has one fixed kernel of equal weight $n^{-1/2}$ and (n^2-1) AC kernels that are derived by the principal component analysis (PCA). We refer to [21] for more details about the Saab transform.

channel. Then, PCA is applied to each channel. Type 5 representation is derived from the original Laws response and PCA coefficients.

3.2. Module 2: Supervised Feature Learning

The total number of representations obtained from Module 1 is around 1500. A subset of the most relevant representations can be selected based on training data to feed into the regressor. This work adopts a mechanism called the relevant feature test (RFT) [22] to achieve this objective. Since RFT is relatively new, it is briefly reviewed below. Let $[f_{min}^i, f_{max}^i]$ denote the value range of the i^{th} representation, and partition the samples by a certain value t ($f_{min}^i \leq t \leq f_{max}^i$) in the i^{th} representation into two nonoverlapping subsets, denoted by S_L^i and S_R^i . Let y_L^i and y_R^i be the mean of target values in S_L^i and S_R^i . They are used as the estimated regression values of all samples in S_L^i and S_R^i , respectively. The RFT loss is defined as the sum of the estimated regression MSEs of S_L^i and S_R^i . Mathematically, it is in the form of

$$R_t^i = \frac{N_{L,t}^i R_{L,t}^i + N_{R,t}^i R_{R,t}^i}{N},\tag{1}$$

where $N_{L,t}^i$, $N_{R,t}^i$, $R_{L,t}^i$, and $R_{R,t}^i$ are the sample numbers and the estimated regression MSEs in subsets S_L^i and S_R^i , respectively, and $N=N_{L,t}^i+N_{R,t}^i$. The RFT loss function of the i^{th} representation is defined as the optimized estimated regression MSE over the set, T, of all candidate partition points, i.e.,

$$R_{op}^i = \min_{t \in T} R_t^i. \tag{2}$$

The lower the RFT loss, the better the representation. We compute the RFT loss values of all representations and sort them in ascending order to yield an RFT loss curve. The elbow point is considered to select a subset of representations with lower RFT loss. This set defines the relevant features to be fed into a regressor in Module 3.

3.3. Module 3: Supervised Decision Learning

In the training process, data augmentation is performed (via 90-degree rotations and flipping) to enlarge the training sample size, and the following two options, "clustering" and "prediction" fusion, are considered. *Clustering*. Perform K-means clustering on ILR patches based on their HOG features and then train the XGBoost regressor in each cluster using features selected from Module 2. *Prediction Fusion*. Augment each ILR patch for multiple times, perform the regression of each one, and take the average of all prediction results as the ultimate predicted residual value.

4. EXPERIMENTS

4.1. Experimental Setup

The SR experiments are conducted with a scaling factor of 2 and the BSD200 dataset [24] is adopted to train the proposed LSR model. The tests are performed on four datasets: Set5 [25], Set14[26], BSD100 [24], and Urban100 [6]. Following the standard routine, we only process the Y channel for super resolution. Our model uses the 15×15 ILR patches to predict the residual value of the center pixel of the patch. An ILR patch of 16×16 is also obtained from the 15×15 ILR patch by the Lanczos interpolation for HOG feature extraction. Table 1 shows statistics and the parameter setting for easy and hard data samples with notations RT_h and FU_h to

Table 1. Statistics and parameter settings for easy and hard data.

Data Type	Easy	Hard
Ratio	56%	44%
Patch variance Range	≤180	≥180
Pixel initial MSE(ILR, HR)	21.78	158.05
Representation Types	[1, 3]	RT_h
Cluster Number	1	8
XGBoost Regressor Tree Number	50	500
XGBoost Regressor Max Depth	6	6
Fusion	No	Yes (FU_h)

Table 2. Average PSNR/SSIM with different settings of representation types (RT) for hard data tested on Set5 and Set14.

PSNR / SSIM	RT_h =[1]	RT_h =[2]	RT_h =[3]
Set5	35.24 / 0.9523	36.12 / 0.9576	36.16 / 0.9578
Set14	31.32 / 0.9074	31.94 / 0.9133	31.96 / 0.9134
PSNR / SSIM	RT_h =[4]	RT_h =[5]	RT_h =[1,2,3,4,5]
Set5	36.11 / 0.9573	36.26 / 0.9583	36.34 / 0.9586
Set14	31.91 / 0.9130	32.04 / 0.9141	32.11 / 0.9147

represent various representation types and fusion schemes for hard samples.

Since the initial MSE of easy data is small, we adopt a simple procedure to predict the residual values of easy data. For hard data, several different settings are compared. First, the PSNR/SSIM performance of different representation types and the union of all five types are demonstrated in Table 2. Representation types 2-5 outperform representation types 1 by a clear margin, while the union of all five types gives the best performance. Here the RFT is applied to the union of all five representation types below to maintain high performance. However, one can choose a single representation type such as type 5 alone to lower the computational complexity.

Next, we compare the performance of four different decision schemes: 1) $FU_h = 1$: without clustering and prediction fusion, 2) $FU_h = 2$: with clustering but no prediction fusion, 3) $FU_h = 3$: with both clustering and prediction fusion (fusion by 2), and 4) $FU_h = 4$: with both clustering and prediction fusion (fusion by 4). The results are shown in Table 3, which confirm the effectiveness of clustering and prediction fusion.

4.2. Quality Performance Comparison

Table 4 demonstrates the quality comparison of two versions of the proposed LSR method (V1: RT_h =[1,2,3,4,5], FU_h =3, and V2: RT_h =[5], FU_h =3) and three light-weight SR methods (SelfExSR[6], A+ [5], and SRCNN[2]). Note that SRCNN is the simplest DL-based method. Here we do not include advanced DL-based solutions in the table since their model sizes and computational complexity are too high to be used on mobile/edge platforms. This table exhibits that LSR V1 and LSR V2 achieve the best SSIM performance among all benchmarking methods while their PSNR performance is close to SRCNN.

We also show four test SR images in Fig. 3 for visual comparison. LSR V1 and SRCNN offer better visual quality with shaper edges and textures than SelfExSR and A+. Although the visual quality of LSR and SRCNN is comparable, their complexity is quite different as presented in the following subsection.

Table 3. Average PSNR/SSIM with different fusion schemes for hard data against Set5 and Set14.

mara aana ag	annot bette and	Sect		
PSNR/SSIM	FU_h =1	FU_h =2	FU_h =3	FU_h =4
Set5	36.33 / 0.9588	36.49 / 0.9592	36.57 / 0.9595	36.60 / 0.9597
Set14	32.12 / 0.9150	32.25 / 0.9156	32.30 / 0.9159	32.32 / 0.9161

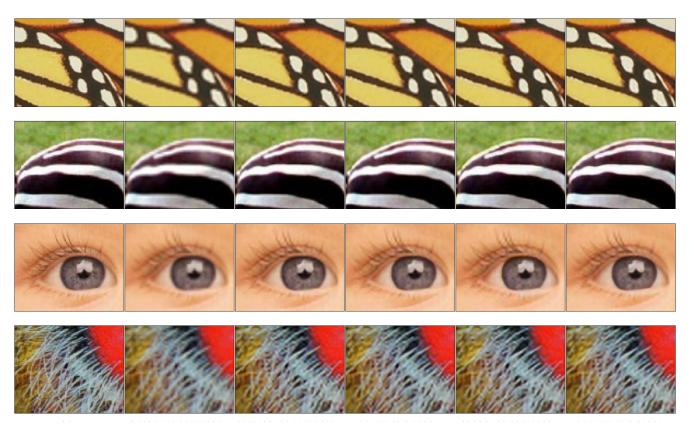


Fig. 3. Visual comparison of 4 test SR images, where the PSNR/SSIM results are provided in brackets.

Table 4. Comparison of averaged PSNR/SSIM values on datasets Set5, Set14, B100 and Urban100, where red and blue colors indicate the best and the second best performance for each dataset.

PSNR/SSIM	SelfExSR[6]	A+[5]	SRCNN [2]	LSR (Ours), V1	LSR (Ours), V2
Set5	36.49 / 0.9537	36.54 / 0.9544	36.66 / 0.9542	36.57 / 0.9595	36.50 / 0.9593
Set14	32.22 / 0.9034	32.28 / 0.9056	32.42 / 0.9063	32.30 / 0.9159	32.23 / 0.9155
BSD100	31.18 / 0.8855	31.21 / 0.8861	31.36 / 0.8879	31.37 / 0.8985	31.32 / 0.8980
Urban100	29.54 / 0.8967	29.20 / 0.8938	29.50 / 0.8946	29.51 / 0.9024	29.42 / 0.9013

Table 5. Comparison of computational complexity (FLOPs per pixel) and model sizes of five SR methods.

Complexity	FLOPs / pixel	Model Size
A+[5]	15.7K(4X)	1.06M (18.6X)
SRCNN[2]	114K (30X)	57.3K (1X)
VDSR[9]	1.33M (347X)	665K (11.6X)
LSR (Ours), V1	9.28K (2.42X)	774K (13.51X)
LSR (Ours), V2	3.83K (1X)	770K (13.45X)

4.3. Complexity and Model Size Comparison

The computational complexity is measured in terms of floating-point operations (FLOPs) per pixel in inference, and the model size in terms of the number of model parameters. Three benchmarking methods are compared to the proposed LSR in Table 5. As one of the best non-DL-based method, A+ (1024-atom dictionary version) shows reasonable FLOPs value but large model size. SRCNN (9-5-5 version) has very small model size, while its FLOPs is large. As a median size DL-based method, VDSR shows very large FLOPs value. Although model size of LSR V1 is comparable with VDSR, its FLOPs per pixel is only 0.70% of VDSR. Besides, when achieving similar PSNR/SSIM, The FLOPs per pixel of LSR V1 is only

8.11% of SRCNN. LSR V2 even reduces FLOPs per pixel to 3.35% of SRCNN. Our model LSR shows extremely low inference computational complexity, and its model size is also acceptable for mobile devices.

5. CONCLUSION AND FUTURE WORK

A light-weight SR method, called LSR, was proposed in this work. It offers good visual quality that is comparable with that of SRCNN, which is an entry-level DL-based method, but at a significantly lower computational complexity (i.e., 8.11% in terms of FLOPs per pixel by V1, even 3.35% by V2). Besides, we presented a wide range of design choices that can lower the computational cost even more with slight quality degradation (see Table 2 and Table 4).

There are several topics worth our future research. First, we would like to generalize our current method to other scale factors ($\times 3$, $\times 4$, etc.). Second, it is desired to boost visual quality furthermore while keeping low computational complexity, which can be achieved by more effective ensemble learning. Third, it is critical to develop a real-time SR video solution. The main challenge from image-based to video-based SR is preservation of temporal smoothness.

6. APPENDICES

The detailed calculations on the computational complexity in inference (in terms of FLOPs) and the model size (in terms of the number of model parameters) of the methods involved in Table 5 are presented in this section. The image "woman.png" of resolution 344×228 in the Set5 test dataset is used as an example. All the calculation is based on the original codes published by authors of each paper. "FLOPs per pixel" in each step is obtained by dividing FLOPs of the whole image by the pixel number in the final predicted HR. FLOPs, FLOPs per pixel, and model sizes are denoted by F, F_p and M, respectively.

Interpolation from low resolution images to the same size of high resolution images is commonly adopted as a pre-processing step in all algorithms of consideration. Since the interpolation process is usually fast and no learned model required, our complexity computation below does not involve this procedure. Besides, different algorithms have different strategies in handling image borders. For fair comparison, we assume that all algorithms generate feature maps and predict HR image based on the ILR image.

6.1. FLOPs and Model Size of Typical Operations

There are several typical procedures involved in various SR algorithms. The calculation of F and M on these procedures is discussed below.

Pixel-wise operation. For a single pixel-wise operation (addition or multiplication) on a set of number N images (or patches) with height H, width W, and depth (number of channels) C, we have $F = H \times W \times C \times N$.

Matrix Multiplication. For a matrix multiplication between a $T_h \times T_w$ transform matrix and $T_w \times N$ sample matrix for N samples, we have

$$F = (2 \times T_w - 1) \times T_h \times N, \tag{3}$$

$$M = T_h \times T_w, \tag{4}$$

with T_w multiplications and (T_w-1) additions for each element in the $T_h\times N$ output matrix.

3D Convolution or Filtering. For the convolution operations that generate 2D feature maps by 3D convolution kernels, we use C_i to denote the number of channels of the input image, and K_h and K_w to represent height and width of the convolution kernel, respectively. To generate one spatial feature response in one feature map, we need

$$2 \times C_i \times K_h \times K_w - 1$$

operations, including $C_i \times K_h \times K_w$ multiplications and $C_i \times K_h \times K_w - 1$ additions. The bias-adding operation demands one addition operation for each spatial point at a feature map. Thus, F and M for a 3D convolution operation without bias on an image can be computed as

$$F = (2 \times C_i \times K_h \times K_w - 1) \times H_o \times W_o \times C_o, \quad (5)$$

$$M = (C_i \times K_h \times K_w) \times C_o, \tag{6}$$

where H_o and W_o are the height and width of the feature maps, respectively, and C_o is the number of feature maps. If there exists a bias term, we have

$$F = (2 \times C_i \times K_h \times K_w) \times H_o \times W_o \times C_o, \qquad (7)$$

$$M = (C_i \times K_h \times K_w + 1) \times C_o. \tag{8}$$

Table 6. Calculation of FLOPs (F), FLOPs per pixel (F_p) and model size (M) for A+.

ILR Feature Extraction (IFE)										
Filter Type	C_i	K_h	K_w	H_o	W_o	C_o	F	F_p	M	
D_w^1	1	1	3	344	228	1	0.39M	5.00	3	
D_h^1	1	3	1	344	228	1	0.39M	5.00	3	
D_w^2	1	1	5	344	228	1	0.71M	9.00	5	
$D_h^{\widetilde{2}}$	1	5	1	344	228	1	0.71M	9.00	5	
IFE Sub-tota	al						2.20M	28.00	16	

Residue Patch Prediction (RPP)									
Step	T_h	T_w	N	F	F_p	M			
ILR Feat. Dim. Red.	28	144	18480	0.15B	1893.43	4032			
Dist. to ILR Atoms	1024	28	18480	1.04B	13270.01	28672			
Regression Prediction	36	28	18480	0.04B	466.52	1032192			
RPP Sub-total				1.23B	15629.96	1064896			

HR Image Prediction (HIP)									
Step	Н	W	С	N	F	F_p	\overline{M}		
Add. ILR to pred. Res.	6	6	1	18480	0.67M	8.48	0		
Cumu. of pixel values	6	6	1	18480	0.67M	8.48	0		
Div. by pixel counter	344	228	1	1	0.08M	1.00	0		
HIP Sub-total					1.41M	17.96	0		

Summary									
Procedure	F	F_p	M						
IFE Sub-total	2.20M	28.00	16						
RPP Sub-total	1.23B	15629.96	1064896						
HIP Sub-total	1.41M	17.96	0						
Total	1.23B	15675.93	1064912						

6.2. A+

As an example-based SISR algorithm, A+ uses 6×6 ILR patches (with overlapping width 2) to predict the the corresponding 6×6 residue patches, and generate the average values for patch overlapping regions. A+ mainly contains three sequential procedures: 1) ILR feature extraction, 2) residue patch prediction, and 3) HR image prediction. The calculation of F, F_p and M of A+ are given in Table 6. Based on the example image, 18480×6 patches are formed in the entire process.

ILR Feature Extraction. Four feature maps are generated using the first and the second order derivative filters (D^1, D^2) along the image height (D^1_h, D^2_h) and width (D^1_w, D^2_w) , respectively.

Residue Patch Prediction. For a 6×6 ILR patch, ILR raw features of 144 dimensions are formed by taking the corresponding 6×6 region in four feature maps and conduct feature concatenation. Afterward, three steps are executed to generate the residue patches: 1) reduce ILR features from 144 dimensions to 28 dimensions, 2) calculate distance to 1024 ILR dictionary atoms to identify the closest atom for each ILR patch, and 3) predict the residue patch values by the regressor associated with the closest ILR dictionary atom for each ILR patch. All the three steps are implemented by 2D matrix multiplication. Then, raw regression predictions (36-D) are reshaped into 6×6 patches to form the eventual residue patch prediction. In the third step, "Regression Prediction", of this procedure, F is calculated by the closest regressor, while M includes regressors associated all 1024 ILR dictionary atoms.

HR Image Prediction. Predicted HR patches are obtained by adding corresponding ILR values to the predicted residue patches. Then, they are used to reconstruct the complete predicted HR image one by one. Two 344×228 all-zero matrices are generated, with one

Table 7. Calculation of FLOPs (F), FLOPs per pixel (F_p) and model size (M) for SRCNN.

Steps	C_i	K_h	K_w	H_o	W_o	C_o	F	F_p	M
conv1	1	9	9	344	228	64	0.81B	10368	5248
conv2	64	5	5	344	228	32	8.03B	102400	51232
conv3	32	5	5	344	228	1	0.13B	1600	801
Total							8.97B	114368	57281

Table 8. Calculation of FLOPs (F), FLOPs per pixel (F_p) and model size (M) for VDSR.

Steps	C_i	K_h	K_w	H_o	W_o	C_o	N_l	F	F_p	M
conv1	1	3	3	344	228	64	1	90.35M	1152	576
conv2 - 19	64	3	3	344	228	64	18	104.09B	1327104	663552
conv20	64	3	3	344	228	1	1	90.35M	1152	576
post-process				344	228	1	1	78.43k	1	0
Total								104.29B	1329409	664704

for pixel value accumulation, and the other for counting pixel coverage times from different patches. The final HR image prediction is obtained by the division of the accumulation matrix by the counting matrix. All steps in this procedure are pixel-wise operations.

6.3. SRCNN

SRCNN is a DL-based method that has three convolution layers with bias on ILR images to generate predicted HR images. The calculation of F, F_p and M are shown in Table 7.

6.4. VDSR

VSDR is a 20-layer DL-based method with 3×3 kernel for each layer. VDSR utilizes ILR images to predict residue images, and final predicted HR images are obtained by adding predicted ILR residues to the ILR images. The calculation explanation of F, F_p and M are exhibited in Table 8, with N_l denotes number of layers.

6.5. LSR

Since inference samples are partitioned into easy and hard samples, we compute FLOPs per pixel, F_p , using the weighted sum of easy and hard samples, where the weight is determined by the ratio of easy and hard samples in representative images. The model size includes the model parameters in both partitions. F_p and M calculations for each module are provided in Table 9. The complexity calculation for V1 and V2 are, respectively, summarized in Table 10.

Module 1: Unsupervised Representation Learning (URL). Being slightly different from the 3D convolution operation without bias (eq. 5) using 3D kernels or filters, LSR uses channel-wise 2D filters. RT in Table 9 denotes the representation type(s). F_p and M of LSR in Module 1 for a certain representation type are obtained by

$$F_p = C_i \times (2 \times K_h \times K_w - 1) \times C_o \times N_{type}, \qquad (9)$$

$$M = C_i \times (K_h \times K_w) \times C_o \times N_{type}, \tag{10}$$

where N_{type} counts the type number regarding to original filter or PCA filters for a certain representation type, which is typically used for Type 4 and 5 Representation. $N_{type}=0$ means no filtering operation needed, nor is filter parameter needed to store. $N_{type}=1$ means the normal transform operations and corresponding filters are required. $N_{type}=2$ means both normal transform and channel-wise PCA transform involved. We use f to denote inference augmentation

Table 9. Calculation of FLOPs per pixel (F_p) and model size (M) for LSR in each module.

Module 1: Unsupervised Representation Learning (URL)

RT	C_i	K_h	K_w	C_o	N_{type}	F_p	M
Type 1, Spatial	1	1	1	1	0	0	0
Type 2, Central Saab	1	5	5	25	1	1225	625
Type 2, Central Saab	1	7	7	49	1	4753	2401
Type 3, Ringwise Saab	1	3	3	9	1	153	81
Type 4, Haar & PCA	1	2	2	4	2	56	32
Type 5, Laws & PCA	1	3	3	9	2	306	162
LIRL Sub-total		RT	F_{r}	/f f	F_n		

URL Sub-total	RT	F_p/f	f	F_p	M
Easy	[1, 3]	153	1	153	81
Hard (V1)	[1, 2, 3, 4, 5]	6493	2	12986	3301
Hard (V2)	[5]	306	2	612	102

Module 2: Supervised Feature Learning (SFL)

RFT Sub-total	RT	N_{fr}	F_p	\overline{M}
Easy	[1, 3]	105	0	105
Hard (V1)	[1, 2, 3, 4, 5]	374	0	374
Hard (V2)	[5]	135	0	135

Module 3: Supervised Decision Learning (SDL)

Cluste	r Pred.	N_{fc}		N_c	F_{I}	, / f	f	F	p	M
Easy		0		1		0	1	()	0
Hard		32		8	7	60	2	15	20	256
Regress	or Pred.	N_{tree}	d_M	F_p/f	f	F_p	M/clu	ster	N_c	M
Easy		50	6	300	1	300	9500)	1	9500
Hard		500	6	3000	2	6000	9500	0	8	760000
-	Predi	iction Fu	sion		f		F_p	1	M	_
	Easy				1		0		0	_
	Hard				2		2		0	

times for fusion. The sibling candidate samples for one inference sample undergo the complete prediction process. Thus, F_p for one inference sample needs to consider the f factor.

Module 2: Supervised Feature Learning (SFL). F in this module is zero due to absence of mathematical operations. The model stores the representation indices which are selected as regression features. Thus, M is determined by the number of features for regression (N_{fr}) selected from representation pool.

Module 3: Supervised Decision Learning (SDL). There are three steps for inference samples in this module: cluster prediction, regressor prediction, and prediction fusion.

• Cluster Prediction. Denote the number of clustering feature by N_{fc} , and cluter number by N_c . For one sample, the FLOPs consumed in its cluster label prediction derives from the calculation of L2 distance to all N_c cluster centroids, which can be simplified as first term in eq. (11).

$$F_p = max((3 \times N_{fc} - 1) \times N_c, 0) \times f, \qquad (11)$$

including N_{fc} subtraction, N_{fc} multiplication, and $(N_{fc}-1)$ addition operations with respect to each cluster centroid. The maximum operation in eq. (11) is for the calculation generalization for easy data without clustering procedure. F_p for each inference sample involves the multiplication by f number of sibling candidate samples. M only contains all cluster centroids in the clustering procedure. It is equal to

$$M = N_{fc} \times N_c. \tag{12}$$

Table 10. Calculation of FLOPs and model size for VDSR.

V1 Summary

, 1 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1							
Complexity	F_p		M				
Data Type	Easy	Hard	Easy	Hard			
URL Sub-total	153	12986	81	3301			
SFL Sub-total	0	0	105	374			
SDL Sub-total	300	7522	9500	760256			
Post-process	1	1	0	0			
Sub-Total	454	20509	9686	763931			
w	0.56	0.44	1	1			
Total	92	278	773617				

V2 Summary

Complexity	F	\vec{p}	M		
Data Type	Easy	Hard	Easy	Hard	
URL Sub-total	153	612	81	162	
SFL Sub-total	0	0	105	135	
SDL Sub-total	300	7522	9500	760256	
Post-process	1	1	0	0	
Sub-Total	454	8135	9686	760553	
w	0.56	0.44	1	1	
Total	38	34	770239		

• Regressor Prediction. LSR learns an XGBoost regressor in each cluster for prediction. The upper bound of FLOPs of one sample prediction by a XGBoost regressor with N_{tree} number of boosting trees and maximum depth d_M is calculated by $d_M \times N_{tree}$, where d_M is the FLOPs value for one boosting tree, as one sample at most traverses d_M nodes until it arrives at one leaf node, and one node only performs one operation. Similar to the clustering procedure, FLOPs for one inference sample also need the multiplication by fusion number f. Thus, we have

$$F_p = d_M \times N_{tree} \times f, \tag{13}$$

For a complete binary decision tree with depth d_M , the numbers of leaf nodes (N_{leaf}) and parent nodes (N_{parent}) are calculated by

$$N_{leaf} = 2^{d_M}, N_{parent} = \sum_{d=1}^{d_M-1} 2^d = 2^{d_M} - 1,$$
 (14)

and its parameter number is $(2 \times N_{parent} + N_{leaf})$, with parent nodes storing feature index and partition threshold, and leaf nodes storing the prediction weight. Thus, the value of M for all regressors is bounded by

$$M = (2 \times N_{parent} + N_{leaf}) \times N_{tree} \times N_c.$$
 (15)

Prediction Fusion. Hard data need additional (f - 1) additional operation and one division operation to average the predictions from sibling samples.

The raw regression results are residue pixel values. We need a pixel-wise post-processing step that adds a residual to a ILR value to yield the ultimate HR prediction. The difference between V1 and V2 models lies in the URL representation preparation and SFL feature selection.

7. REFERENCES

- [1] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical models and image processing*, vol. 53, no. 3, pp. 231–239, 1991.
- [2] C. Dong, C. C. Loy, K. He, and X. Tang, "Image superresolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.
- [3] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004. CVPR 2004., vol. 1. IEEE, 2004, pp. I–I.
- [4] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE transac*tions on image processing, vol. 21, no. 8, pp. 3467–3478, 2012.
- [5] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Asian conference on computer vision*. Springer, 2014, pp. 111–126.
- [6] J.-B. Huang, A. Singh, and N. Ahuja, "Single image superresolution from transformed self-exemplars," in *Proceedings* of the IEEE conference on computer vision and pattern recognition, 2015, pp. 5197–5206.
- [7] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3791–3799.
- [8] P. Sandeep and T. Jacob, "Single image super-resolution using a joint gmm method," *IEEE Transactions on Image Processing*, vol. 25, no. 9, pp. 4233–4244, 2016.
- [9] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image superresolution using very deep convolutional networks," in *Pro*ceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 1646–1654.
- [10] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2472–2481.
- [11] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 11065–11074.
- [12] Y. Mei, Y. Fan, Y. Zhou, L. Huang, T. S. Huang, and H. Shi, "Image super-resolution with cross-scale non-local attention and exhaustive self-exemplars mining," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5690–5699.
- [13] H. Chen, Y. Wang, T. Guo, C. Xu, Y. Deng, Z. Liu, S. Ma, C. Xu, C. Xu, and W. Gao, "Pre-trained image processing transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12299– 12310.
- [14] X. Kong, H. Zhao, Y. Qiao, and C. Dong, "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*, 2021, pp. 12016–12025.

- [15] L. Wang, X. Dong, Y. Wang, X. Ying, Z. Lin, W. An, and Y. Guo, "Exploring sparsity in image super-resolution for efficient inference," in *Proceedings of the IEEE/CVF conference* on computer vision and pattern recognition, 2021, pp. 4917– 4926
- [16] R. Zhou and S. Susstrunk, "Kernel modeling super-resolution on real low-resolution images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2433–2443.
- [17] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Metasr: A magnification-arbitrary network for super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1575–1584.
- [18] C.-C. J. Kuo and A. M. Madni, "Green learning: Introduction, examples and outlook," arXiv preprint arXiv:2210.00965, 2022.
- [19] C.-C. J. Kuo, "Understanding convolutional neural networks with a mathematical model," *Journal of Visual Communication* and *Image Representation*, vol. 41, pp. 406–413, 2016.
- [20] C.-C. J. Kuo and Y. Chen, "On data-driven saak transform," Journal of Visual Communication and Image Representation, vol. 50, pp. 237–246, 2018.
- [21] C.-C. J. Kuo, M. Zhang, S. Li, J. Duan, and Y. Chen, "Interpretable convolutional neural networks via feedforward design," *Journal of Visual Communication and Image Representation*, vol. 60, pp. 346–359, 2019.
- [22] Y. Yang, W. Wang, H. Fu, C.-C. J. Kuo et al., "On supervised feature selection from high dimensional feature spaces," APSIPA Transactions on Signal and Information Processing, vol. 11, no. 1, 2022.
- [23] K. I. Laws, "Rapid texture identification," in *Image processing* for missile guidance, vol. 238. Spie, 1980, pp. 376–381.
- [24] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2. IEEE, 2001, pp. 416– 423.
- [25] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," 2012.
- [26] R. Zeyde, M. Elad, and M. Protter, "On single image scaleup using sparse-representations," in *Curves and Surfaces: 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers 7.* Springer, 2012, pp. 711–730.