Classification-based Dynamic Network for Efficient Super-Resolution

Qi Wang, Weiwei Fang, Meng Wang, Yusong Cheng
Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University, Beijing, China
{20120417, fangww, 21120407, 21120335}@bjtu.edu.cn

***Abstract*—Deep learning based approaches have achieved superior performance in single image super-resolution (SR). To obtain better visual quality, deep neural networks (DNNs) for SR are generally designed with massive computation overhead. In order to accelerate network inference under resource constraints, we propose a classification-based dynamic network for efficient super-resolution (CDNSR), which combines the classification and SR networks in a unified framework. Specifically, CDNSR decomposes a large image into a number of image-patches, and uses a classification network to categorize them into different classes based on the restoration difficulty. Each class of image-patches will be handled by the SR network that corresponds to the difficulty of this class. In particular, we design a new classification loss to trade off between the computational overhead and the quality of reconstructed image, so as to strike a more favorable accuracy-resource balance. Besides, we apply contrastive learning based knowledge distillation to guarantee the performance of SR networks and the PSNR quality of reconstructed images. Extensive experiments demonstrate that our CDNSR significantly outperforms the other SR networks and backbones in terms of image quality and computational overhead.**

***Index Terms*—Deep Learning, Super Resolution, Adaptive Inference, Knowledge Distillation.**

I. Introduction

The goal of single image super-resolution (SR) is to reconstruct a high-resolution (HR) image from a single low-resolution (LR) images. Recently, many efforts have been made on the real-world applications, e.g., blind SR [1], [2] and arbitrary scale SR [3], [4]. Due to the powerful feature representation and model fitting capabilities of deep neural networks (DNNs), DNN-based approaches have achieved significant performance improvements over conventional methods [5]. Since the seminal work of SRCNN [5], the DNN architectures for image SR become deeper and deeper. For example, EDSR [6] proposes a deep and wide backbone with more than 60 layers. The network depth in two following work, i.e., RDN [7] and RCAN [8], increases to over 100 and 400, respectively.

With the proliferation of smart edge devices such as smartphones and VR glasses, SR is in high demand in human daily-life. However, the tension between resource-hungry DNN models and resource-poor edge devices adversely affects user experiences, and inspires new research efforts on efficient SR. To address this issue, researches have been made on feature reuse [9] and information distillation [10], but they still involve redundant computations.

This is the technique report for CDNSR, which has been accepted by ICASSP 2023.

Compared with HR images, the missing details in its LR images mainly exist in the edge and texture regions, while fewer computing resources are required for smooth regions. However, current DNN-based SR solutions treat all the regions indiscriminatingly, leading to redundant computations for smooth regions. Inspired by two efficient DNN designs, i.e., DRNet [11] and ClassSR [12], we propose a new SR framework CDNSR (Classification-based Dynamic Network for efficient Super-Resolution) that feeds different SR networks with the decomposed image-patches with different restoration difficulties, so as to balance the trade-off between performance and cost. There are two main modules in CDNSR. On the one hand, the classification module is a simple neural network that categorizes the input into a specific class according to its restoration difficulty. We design the loss by taking both image classification and computational cost into consideration, and adopt Gumbel Softmax trick [11] to solve the non-differentiable problem in the process from the classifier's continuous outputs to discrete class selections. On the other hand, the SR module is a container that processes the classified input with the corresponding SR network. To avoid performance degradation, knowledge distillation is adopted to transfer information from the large SR network (teacher) to the smaller one (student). We introduce a new contrastive loss that provides to the student network not only an upper bound to enforce the output closer to the teacher [13], but also a lower bound to enforce the output farther away from negative images in the feature space [14]. The CDNSR framework is general and effective for different SR backbones. Fig. 1 shows the results on representative SR backbones, including FSRCNN (tiny scale) [15], CARN (small scale) [9], SRResNet (medium scale) [16], and RCAN (large scale) [7].

Our main contributions can be summarized as follows:

1) The proposed CDNSR is a universal framework that implements efficient SR at the image-patch level. In contrast to existing studies that focus on light-weight backbone design [17], [18], we explore a different instance-based computation manner to improve inference efficiency.
2) CDNSR can achieve a good trade-off between computational efficiency and SR performance. On the one hand, it selects appropriate SR networks to process the image-patches with different restoration difficulties for saving computational cost. On the other hand, it

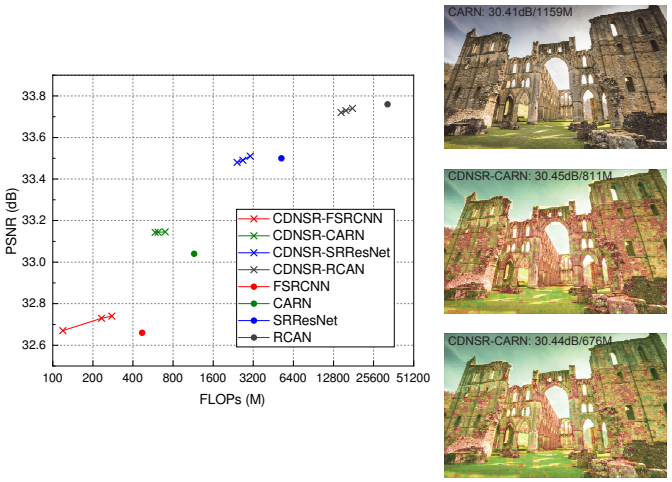

Fig. 1. Left: PSNR and FLOPs comparisons between CDNSR and original backbones on Test8K dataset with 4×. Right: Visual results of CDNSR and the original backbones. Note that the image-patches are classified into simple (green), medium (yellow), and hard (red) classes.

adopts contrastive learning based knowledge distillation to constraint the output of SR networks within the upper and lower bounds for maintaining image quality.

3) Extensive experiments show that our CDNSR can accelerate various popular SR backbones (e.g., FSRCNN, CARN, SRResNet, RCAN) with superior performance. Besides, it can achieve comparable or better performance than ClassSR in most cases.

II. Related Work

A. Efficient Image Super-Resolution

Pioneered by the 3-layer network SRCNN [5], the research on single-image SR has been dominated by DNN-based approaches, because of their powerful representation and fitting capabilities. For instance, EDSR [6] has a total of over 60 convolutional layers by cascading modified residual blocks, and RDN [7] is built with more than 100 convolutional layers by combining residual blocks and dense connections. RCAN [7] has as many as 200 residual blocks and more than 400 convolutional layers. While these DNNs can achieve a very high performance, the expensive computational cost has hindered their practical deployments and applications, especially on resource-constrained edge devices.

To address the efficiency issue, many approaches have been adopted to reduce model complexity. Both FSRCNN [15] and ESPCN [19] take LR images as input and employ post-upsampling layers to reduce computational burden. CARN [9] saves the network parameters by adopting group convolution and cascading connections for learning multi-level feature representations. IMDN [17] is a light-weight multi-distillation network that applies feature splitting and concatenation operations for feature aggregation. PAN [18] proposes pixel attention to help the light-weight backbone generate acceptable restoration results.

However, these works aim to design a dedicated light-weight network with a satisfactory restoration performance. By comparison, our proposed CDNSR is a universal framework that can work with most existing SR backbones to help them achieve a flexible performance-cost trade-off.

B. Knowledge Distillation

Knowledge distillation is a learning framework that transfers information to a compact network (student) from a more capable, but cumbersome network (teacher). This idea of model compression was first proposed by Hinton *et al.* [20], in which the outputs of student and teacher are matched by minimizing the KL-divergence of the category distribution. Some works focus on what knowledge to distill, including response-based knowledge [20], feature-based knowledge [21], and relation-based knowledge [22]. Another line of studies pay more attention to how to distill the knowledge, including offline/online distillation [20], [23].

Contrastive learning is a self-supervised learning technique that learns the general features of input samples without labels, by letting similar sample pairs stay close to each other and dissimilar ones stay far apart. It has already been employed on image-to-image translation [24] and image dehazing [25]. Recently, the contrastive loss is introduced to explicitly represent knowledge in distillation and provide the closed upper and lower bounds to improve the model performance. For instance, SSKD [26] proposes to transfer cross-sample contrastive relationships for classification tasks, and CSD [14] designs a new contrastive loss in the teacher-student training process for SR tasks.

III. Methodology

In this section, we first provide an overview of our CDNSR framework in section III-A, including Module-CL for classification and Module-SR for SR. Section III-B introduces the loss function and Gumbel Softmax trick in the design of Module-CL, and Section III-C describes the contrastive loss based knowledge distillation for improving the performance of Module-SR. At last, the overall training strategy is given in Section III-D.

A. Overview

As shown in Fig. 2, our proposed CDNSR consists of a Module-CL and a Module-SR. Module-CL classifies the input image-patches into a total of M classes with a classification network, and Module-SR contains M SR-Nets to process different inputs correspondingly. Specifically, the input LR image is firstly decomposed into a number of overlapping image-patches. Module-CL accepts each image-patch, and generates a probability vector $P = [p_1, p_2, ..., p_M]$ as the class predictor. Then, Module-SR determines which SR-Net to be used to process this patch, by selecting the one of the class corresponding to the highest probability entry in P. After being processing by different SR-Nets, all the output image-patches are combined to obtain the final SR image, where the overlapped area is obtained through averaging.

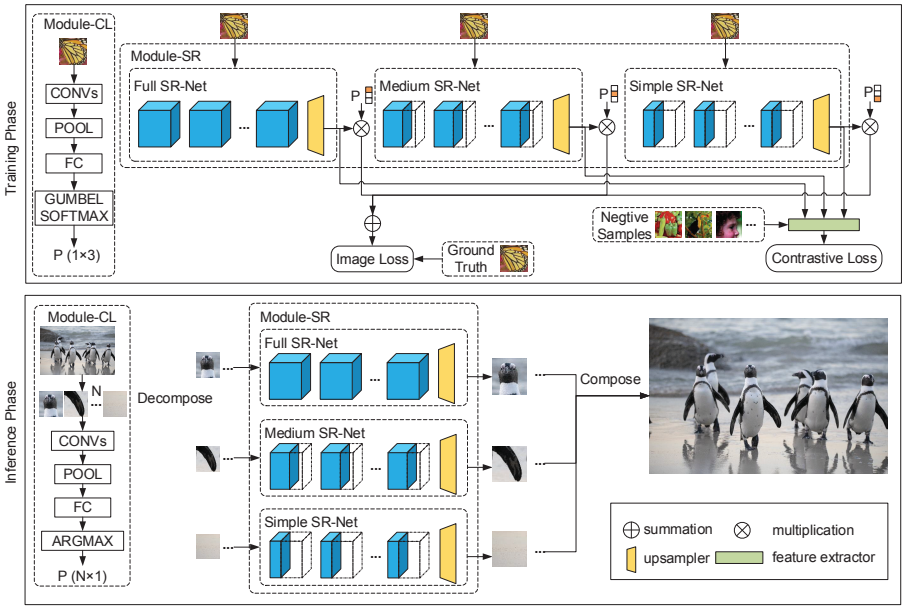

Fig. 2. The overview of the proposed CDNSR, with the number of classes $M = 3$. In the Training Phase, the Image Loss is firstly adopted for individual training, and then the Contrast Loss is adopted as the distillation loss for joint training. In the Inference Phase, the image is clipped, and the final SR image is obtained by combining the sub patches generated by different SR networks.

1) Module-CL.: Module-CL aims to determine the difficulty level for reconstructing the input image-patch based on low-level feature. Inspired by DRNet [11], we design the Module-CL as a simple classification network. It consists of a small number of convolutional layers for feature extraction, followed by an average pooling layer and a fully-connected layer for outputting the probability results. Note that, to support end-to-end training, we adopt the Gumbel Softmax trick [27] to replace $\arg\max$ with a softmax during the training process. Though very simple, we can observe from experiments results that this structure can obtain satisfactory classification outputs while incurring little extra computational cost.

2) Module-SR.: Module-SR aims to select an appropriate SR-Net from a number M of candidates to reduce the computational cost for image-patches. Since we hope to accelerate an existing SR backbone (e.g., CARN), we treat the target backbone as the base SR-Net which is the most complex. Then, we obtain the other networks by slimming the base SR-Net to different extent, by borrowing the idea of channel pruning in DNN compression [28]. The smaller a SR-Net, the easier the input image-patches it will deal with. The pruning principle is that, when trained on the image dataset corresponding to its class, the current small SR-Net has to achieve comparable performance with the base one. For example, in our experiments the number of channels for CARN variants are 32, 56, and 64, respectively.

B. Optimized Design for Module-CL

The CDNSR framework is optimized to perform the instance-aware classification with end-to-end training, so as to match the image-patch with the SR-Net. The loss function and Gumbel Softmax trick for Module-CL are introduced as follows.

1) Loss Function.: The loss function for Module-CL is consisted of two parts: the image restoration loss (i.e., ℓ_1 loss function) L_{res} and a FLOPs constraint regularization L_{reg} for computation budget restriction. The ℓ_1 loss measures the distance between the output image and ground truth y [6], [7], and is defined as:

$$L_{res} = \ell_1(\hat{y}, y) = \ell_1(\sum_{j=1}^{M} p_j y_j, y) \ , \tag{1}$$

where y_j denotes the reconstructed results of j-th SR-Net. Note that we add a regularization on FLOPs to guide the learning of Module-CL [11]. The reason is that, if only the ℓ_1 loss is considered, the classifier will tend to always select the class corresponding to the the base SR-Net because it generates relatively lower restoration loss generally. However, we know that a smaller SR-Net may also be capable enough to deal with simple inputs with fewer FLOPs. In order to reduce the computational cost, we enforce a penalty by the regularization on FLOPs, which is defined as:

$$F = \sum_{j=1}^{M} (C_j * p_j), \tag{2}$$

$$L_{reg} = \max\left(0, \frac{\mathbb{E}(F)}{C_{max}} - \alpha\right) \ , \tag{3}$$

where F denotes the actual FLOPs, C_j and C_{max} denote the pre-computed FLOPs for the j-th SR-Net and the base SR-Net, and α denotes the target FLOPs constraint, respectively.

Finally, the overall lost L_{CL} is defined as the weighted summation over L_{res} and L_{reg} as:

$$L_{CL} = L_{res} + \lambda_{reg} L_{reg} \ , \tag{4}$$

where λ_{reg} is a hyper-parameter to match the magnitude of these two losses.

2) Gumbel Softmax Trick.: During training, there exists a challenge that the discrete output of Module-CL is non-differentiable and can not be back-propagated end-to-end. This is solved by adopting the Gumbel Softmax trick introduced in [27]. Specifically, the discrete candidate SR-Net selections can be drawn as:

$$x = \text{one\_hot}[\arg\max_{j}(\log p_j + g_j)], \quad j \in [1, ..., M] \ , \tag{5}$$

where $g_j = -\log(-\log U_j)$ is Gumbel noise with U_j sampled from a uniform distribution $U(0, 1)$. We can use Gumbel Softmax distribution as a continuous relaxation to $\arg\max$, by defining a relaxed form using the softmax function as:

$$x_j = \frac{\exp(\log p_j + g_j)/\tau}{\sum_{k=1}^{M}(\exp(\log p_k + g_k)/\tau)}, \quad j = 1, \ldots, M \ , \tag{6}$$

where τ is a temperature hyper-parameter that controls the difference between the softmax and $\arg\max$ functions. By this simple trick, the overall framework can be trained end-to-end.

C. Knowledge Distillation for Module-SR

We construct the Module-SR by pruning the base SR-Net to different extent in the channel dimension. Among these SR networks, the larger ones have more capacity, and are generally more accurate than the smaller ones. Knowledge distillation is a natural way to boot the performance of smaller SR-Net by encouraging them to imitate the larger ones [20]. In CDNSR, our knowledge distillation takes into account not only the output of teacher network as the upper bound, but also the negative images in the feature space as the lower bound.

According to [20], the distillation loss in the traditional teacher-student mode can be defined as:

$$L_{KD} = L_{res} + \lambda_{KD} \sum_{j=1}^{M-1} \sum_{k=j+1}^{M} \ell_1(y_j, y_k) \ , \tag{7}$$

where j and k denote the teacher and the student, λ_{KD} denotes a hyper-parameter to match the magnitude of these two losses, respectively. Note that each SR-Net can learn from all those ones larger than itself, so as to fully release the potential capabilities of knowledge distillation [29]. Such a teacher-student mode only pulls the restoration result closer to the HR image. Nevertheless, SR tasks are different from those of image classification, and we also need to push the restoration farther away from negative samples in the representation space.

We adopt contrastive learning [30], [31] to address this issue, by proposing a new contrastive loss to explicitly represent knowledge for training the student. Three aspects are to be considered, i.e., how to construct the positive and negative samples, how to construct a latent feature space for comparing these samples, and how to measure the similarity between sample features. To answer the first question, given an input sample (i.e., anchor z), we treat the outputs from the teachers as positive samples z^{pos}. Meanwhile, we sample S images (other than this anchor) as negative samples $z_1^{neg}, \ldots, z_S^{neg}$, which are upsampled to the same resolution as the anchor by bicubic interpolation [14]. To answer the second question, we take a similar approach as that in CSD [30]. Specifically, we adopt the widely-used VGG-19 (denoted by ϕ) to extracts representation vectors from these samples, and then map these representations to the space by a two-layer Multi-Layer Perceptron (MLP). To answer the third question, we use a cosine similarity in this work. Based on these positive and negative samples, the contrastive loss for all the SR networks can be constructed as:

$$L_{CT} = -\sum_{k=M}^{2} \sum_{j=k-1}^{1} \log \left(\frac{\exp(\text{cosine}(\phi(z_k), \phi(z_j^{pos})))}{\sum_{s=1}^{S} \exp(\text{cosine}(\phi(z_k), \phi(z_s^{neg})))} \right) \ , \tag{8}$$

where j and k denote the teacher and the student, S denotes the number of negative samples, respectively.

Finally, instead of using L_{KD}, we construct a new overall loss L_{SR} by leveraging contrastive loss L_{CL} into restoration loss L_{res}, as follows:

$$L_{SR} = L_{res} + \lambda_{CT} L_{CT}, \tag{9}$$

where λ_{CT} denotes a hyper-parameter to match the magnitude of these two losses. Note that we use different updating policy for the teacher and the student [14]. For stability, the teacher's gradient from the contrastive loss are detached, while only gradients from the reconstruction loss are updated in the distillation. Besides, the normal gradient update is taken for the student.

D. Training Strategy for CDNSR

Generally, we train the CDNSR in four steps: pre-training Module-SR, training Module-CL, distilling Module-SR, fine-tuning the whole model. The underlying reason is that, if we forcibly train the two modules at the same time from scratch, we may obtain a very unstable SR performance with sub-optimal classification results.

To pre-train Module-SR, we have to firstly classify the image-patches with the given dataset into M classes. Specifically, we use a well-trained MSRResNet [32] to generate the SR results, and calculate the PSNR for the image-patches based on their HR-SR pairs [33]. Then, the image-patches are categorized into M classes of sub-datasets according to their PSNR values (i.e., restoration difficulties), e.g., easy (1/3 largest PSNRs), medium (1/3 medium PSNRs), and hard (1/3 smallest PSNRs) when $M = 3$. Finally, we use a ℓ_1 loss to train all the M SR-Nets, each of which is independently trained using its corresponding sub-dataset. Taking Fig. 2 as an example, we train the Full SR-Net with hard samples, the

Medium SR-Net with medium samples, and the Simple SR-Net with easy samples. Although PSNR may not be perfect for difficulty estimation, we believe it is convenient and acceptable to use PSNR for such a purpose.

After pre-training Module-SR, we fix all its parameters, and continue to train the added Module-CL using the loss L_{CL} on all data. This process could make Module-CL be capable to classify the inputs based on restoration difficulty and computation cost. Note that the hyper-parameter τ for Gumbel Softmax is set as a value that gradually decreases with the training iterations [34].

Afterwards, we fix all the parameters of Module-CL, and adopt knowledge distillation to improve the performance of SR-Nets. The overall model is trained with the loss L_{SR}.

Finally, we fine-tune the whole model. In the process of jointing training, Module-CL and Module-SR can use each other's output to refine the performance of itself.

IV. Experiments

A. Experiment Settings

1) Training Data.: We use the 800 training images and 100 validation images in DIV2K dataset [35] to train and validate CDNSR, respectively. For a fair comparison with existing work, we adopt a similar way as in ClassSR [12] to obtain HR images by down-sampling the training images in different scales, and obtain image-patches with size 32×32 cropped from LR images further down-sampled by a factor of 4 from the HR images. Then, these image-patches are equally divided into $M = 3$ classes based on their PSNR values for Module-SR pre-training. Data augmentation is performed in training through random flipping or random rotation.

2) Testing Data.: Following ClassSR [12], we select 300 images (from index 1201 to 1500) from DIV8K dataset [36], and construct three datasets Test2K, Test4K and Test8K with 100 HR images each through down-sampling. Then, the LR images are obtained through down-sampling these HR images by a factor of 4, and further cropped into 32×32 image-patches in testing.

3) Training Details.: We build a light-weight classification network with five convolutional layers as Module-CL. Without loss of generality, the proposed Module-SR can use four types of SR backbone models with different sizes as the backbones, including FSRCNN-56 (tiny) [15], CARN-64 (small) [9], SRResNet-64 (medium) [16], and RCAN-64 (large) [7]. The number 56, 64, 64 and 64 are the width of the corresponding base SR-Nets. We have totally $M = 3$ different networks in the Module-SR. Following [12], the number of channels are $(56, 36, 16)$ for FSRCNN, $(56, 52, 36)$ for CARN, $(64, 52, 36)$ for SRResNet, and $(64, 50, 36)$ for RCAN, respectively. All models are implemented with PyTorch and trained on a NVIDIA 2080Ti GPU server. The models are trained using the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The mini-batch size is 64. The learning rate is adjusted by the cosine annealing technique from the initial value of 2×10^{-4} to the minimum of 10^{-7}. We set the hyper-parameters λ_{reg} and λ_{CT} as 2×10^{-4} and 10^{-3}, respectively. The temperature parameter τ in Gumbel Softmax is annealed using the schedule $\tau = max(0.4, 1 - \frac{current\_iteration}{total\_iterations})$ [34]. To provide a comprehensive evaluation to CDNSR, we control the parameters α and λ_{reg} to train three variants of CDNSR with relaxed (R), moderate (M) and strict (S) requirements on FLOPs. Note that we use the same parameter settings as in [12], [14] for a fair comparison.

4) Inference Stage and Evaluation Metrics.: The PSNR and SSIM between SR and HR images is used as the metric to evaluate restoration performance for all solutions on the three test datasets, i.e., Test2K, Test4K and Test8K. Meanwhile, the computational cost is measured by the average FLOPs for all 32×32 LR image-patches with $\times4$ super-resolution across all datasets.

B. Experiment Results

1) Quantitative Results.: The comparison results on PSNR/SSIM and FLOPs/Time for different SR solutions with four backbones are shown in Table I, which make several observations. Firstly, these results indicate that these backbone networks have redundant computation more or less. Both ClassSR and CDNSR can achieve better or comparable performance with less computation than the original backbone networks. However, ClassSR only has a fixed performance, while CDNSR can be flexibly adjusted with the FLOPs regularization parameters to achieve a satisfactory performance-cost trade-off. Secondly, the three variants of CDNSR have their own characteristics, as compared to ClassSR. With a relaxed requirement on FLOPs budget, CDNSR(R) can obtain slightly higher PSNR/SSIM values than ClassSR. With a moderate requirement, CDNSR(M) can achieve better or at least comparable performance than ClassSR in terms of PSNR and FLOPs. With a restrict requirement, CDNSR(S) can reduce much more computational cost than ClassSR, while maintaining an acceptable PSNR/SSIM level. Thirdly, the performance of CDNSR can be well improved by knowledge distillation, making CDNSR(R) achieve the highest PSNR/SSIM values in most cases.

2) Qualitative Results.: As shown in Fig. 3, we compare CDNSR(S) with other solutions on the quality of images at $\times4$ SR scale. We can observe that, even with the strict requirement on FLOPs, CDNSR can still obtain the satisfactory visual effects as the original backbones. The observable transitions are very natural and smooth, implying that there would be no incoherence between neighboring image-patches in image restoration.

C. Ablation Study

To further investigate the performance of our CDNSR, we conduct ablation study to analyze the effect of FLOPs parameters, Module-CL architecture, knowledge distillation, and the number of negative samples. Without otherwise specified, FSRCNN is selected as our SR backbone in the experiments.

1) Effect of FLOPs parameters.: Table II shows how the two key hyper-parameters α and λ_{reg} in L_{CL} can affect the SR performance of CDNSR. We start by fixing λ_{reg} as 2×10^{-4},

TABLE I
COMPARISON OF DIFFERENT SR SOLUTIONS ON TEST2K, TEST4K AND TEST8K.

Model	Test2K			Test4K			Test8K		
	PSNR/SSIM	FLOPs	Time	PSNR/SSIM	FLOPs	Time	PSNR/SSIM	FLOPs	Time
FSRCNN	25.61/0.7363	468M (100%)	0.374s	26.90/0.7829	468M (100%)	1.549s	32.66/0.8639	468M (100%)	8.932s
ClassSR	25.61/0.7365	311M (66%)	0.321s	26.91/0.7839	286M (61%)	1.302s	32.73/0.8653	238M (51%)	7.218s
CDNSR(R)	25.62/0.7368	360M (75%)	0.334s	26.92/0.7842	326M (70%)	1.349s	32.74/0.8655	278M (59%)	7.493s
CDNSR(M)	25.62/0.7367	299M (64%)	0.319s	26.91/0.7840	277M (59%)	1.284s	32.74/0.8654	232M (49%)	7.115s
CDNSR(S)	25.58/0.7351	244M (52%)	0.301s	26.88/0.7827	234M (50%)	1.230s	32.69/0.8645	119M (42%)	6.874s
CARN	25.95/0.7511	1.15G (100%)	1.892s	27.33/0.7967	1.15G (100%)	7.831s	33.04/0.8718	1.15G (100%)	45.472s
ClassSR	25.95/0.7528	841M (73%)	1.750s	27.34/0.7985	742M (64%)	7.074s	33.13/0.8740	608M (53%)	39.695s
CDNSR(R)	25.96/0.7534	867M (75%)	1.756s	27.35/0.8001	806M (70%)	7.160s	33.15/0.8756	698M (61%)	40.449s
CDNSR(M)	25.95/0.7532	826M (71%)	1.733s	27.34/0.7991	750M (65%)	7.051s	33.15/0.8751	618M (53%)	39.444s
CDNSR(S)	25.94/0.7507	763M (66%)	1.707s	27.34/0.7970	699M (60%)	6.943s	33.15/0.8733	586M (51%)	39.193s
SRResNet	26.19/0.7624	5.20G (100%)	1.232s	27.65/0.8075	5.20G (100%)	5.096s	33.50/0.8789	5.20G (100%)	29.593s
ClassSR	26.20/0.7625	3.62G (70%)	1.094s	27.66/0.8074	3.30G (63%)	4.395s	33.50/0.8798	2.70G (52%)	24.222s
CDNSR(R)	26.21/0.7628	3.89G (74%)	1.103s	27.67/0.8077	3.59G (69%)	4.476s	33.51/0.8803	3.04G (58%)	24.731s
CDNSR(M)	26.20/0.7625	3.55G (68%)	1.075s	27.66/0.8075	3.24G (62%)	4.336s	33.50/0.8797	2.67G (51%)	23.940s
CDNSR(S)	26.19/0.7623	3.16G (61%)	1.042s	27.65/0.8072	2.91G (56%)	4.220s	33.49/0.8795	2.42G (46%)	23.487s
RCAN	26.39/0.7696	32.60G (100%)	24.173s	27.89/0.8145	32.60G (100%)	100.133s	33.76/0.8847	32.60G (100%)	581.526s
ClassSR	26.39/0.7697	21.22G (65%)	21.790s	27.88/0.8137	19.49G (60%)	89.146s	33.73/0.8837	16.19G (50%)	498.078s
CDNSR(R)	26.40/0.7701	22.41G (68%)	21.926s	27.89/0.8139	20.74G (63%)	89.428s	33.74/0.8840	17.76G (54%)	503.622s
CDNSR(M)	26.39/0.7696	20.37G (62%)	21.518s	27.88/0.8134	18.80G (57%	87.737s	33.73/0.8836	15.85G (48%)	494.806s
CDNSR(S)	26.38/0.7691	18.29G (56%)	21.103s	27.87/0.8131	17.11G (52%)	86.611s	33.72/0.8833	14.62G (45%)	489.908s

TABLE II
EFFECT OF HYPER-PARAMETER SETTINGS IN THE FLOPS REGULARIZATION LOSS.

α	λ_{reg}	FLOPs	DIV2K
0.75	2e−4	323M (69.1%)	28.578
0.65	2e−4	304M (64.9%)	28.572
0.55	2e−4	264M (56.4%)	28.556
0.45	2e−4	246M (52.6%)	28.533
0.45	1e−4	275M (58.8%)	28.560
0.45	4e−4	236M (50.4%)	28.506
0.45	7e−4	198M (42.4%)	28.457
0.45	1e−3	185M (39.6%)	28.422

and adjust α from 0.75 to 0.45. It can be observed that the value of average FLOPs and PSNR gradually decreases from 323M to 246M and from 28.578 to 28.533, respectively. Note that when a stringent FLOPs target (e.g., $\alpha = 0.45$) is imposed, it may not be satisfied since we also have to take the quality of restored image into account. Then, we fix α as 0.55, and adjust λ_{reg} in the range of $[10^{-4}, 10^{-3}]$. It can be found that a larger penalty factor will make the image-patch be assigned to a small SR-Net, resulting in relatively lower FLOPs and PSNR.

2) Effect of Module-CL Architecture.: We build four classifier architectures with different number of convolutional layers, as well as select a light-weight DNN model MobileNetV2 as a reference, to verify the efficiency of Module-CL. In the experiments, α is set as 0.8, which is a relaxed requirement on FLOPs. We can notice that the addition of Module-CL will bring about very little extra computational cost to the whole framework, as compared to that of Module-SR. From the PSNR results demonstrated in Table III, it is better to use the model architecture with five convolutional layers for Module-CL. We use this setting in the other experiments.

TABLE III
EFFECT OF THE NETWORK ARCHITECTURE FOR MODULE-CL ON FSRCNN AND CARN.

Backbone	Module-CL	Module-CL FLOPs	Module-SR FLOPs	DIV2K
FSRCNN	3*Conv+1*FC	1.739M	338M (72.3%)	28.570
FSRCNN	4*Conv+1*FC	2.804M	335M (71.6%)	28.573
FSRCNN	5*Conv+1*FC	3.869M	325M (69.4%)	28.578
FSRCNN	6*Conv+1*FC	5.934M	313M (66.8%)	28.575
FSRCNN	MobileNetV2	6.389M	373M (79.7%)	28.551
CARN	3*Conv+1*FC	1.739M	910M (79.1%)	29.164
CARN	4*Conv+1*FC	2.804M	887M (77.1%)	29.167
CARN	5*Conv+1*FC	3.869M	867M (75.5%)	29.172
CARN	6*Conv+1*FC	5.934M	874M (76.0%)	29.168
CARN	MobileNetV2	6.389M	940M (81.8%)	29.162

TABLE IV
EFFECT OF HYPER-PARAMETER λ_{CT} ON FSRCNN.

λ_{CT}	DIV2K	Test2K	Test4K	Test8K
0	28.471	25.571	26.873	32.688
1e−4	28.483	25.575	26.875	32.679
1e−3	**28.493**	**25.583**	**26.884**	**32.692**
1e−2	28.460	25.563	26.867	32.667

3) Effect of contrastive loss parameter: Table IV shows how the key hyper-parameter λ_{CT} in L_{SR} can affect the SR performance of CDNSR. It can be observed from second and third row that contrastive loss is important in our framework. If λ_{CT} is too larger(e.g., $\lambda_{CT} = 1e{-}2$), we can see the performance loss with the contrastive loss.

4) Effect of Knowledge Distillation.: We first compare the selections of loss in knowledge distillation. We compare our contrastive loss against the pixel loss, perceptual loss, and CSD loss [13], [14]. In the experiment, we set λ_{CT} and S as 1 and 5. As shown in Table V, our contrastive loss achieves the best results on PSNR. Note that the ℓ_1-based CSD loss

Fig. 3. Visual comparison of CDNSR(S) with various backbone networks, and other SR solutions with $\times 4$ super-resolution. The image above is from Test8K "1411", and the image below is from Test4K "1366", respectively.

TABLE V
EFFECT OF THE LOSSES FOR KNOWLEDGE DISTILLATION ON FSRCNN.

Module-SR	Distillation Loss	DIV2K	Test2K	Test4K	Test8K
CDNSR(R)	Without KD	28.572	25.619	26.913	32.738
CDNSR(R)	Pixel Loss	28.553	25.592	26.891	32.720
CDNSR(R)	Perceptual Loss	28.521	25.583	26.878	32.699
CDNSR(R)	CSD Loss	28.570	25.615	26.910	32.738
CDNSR(R)	Ours	28.580	25.624	26.923	32.746
CDNSR(M)	Without KD	28.547	25.613	26.912	32.736
CDNSR(M)	Pixel Loss	28.523	25.595	26.891	32.710
CDNSR(M)	Perceptual Loss	28.511	25.582	26.874	32.689
CDNSR(M)	CSD Loss	28.550	25.616	26.910	32.738
CDNSR(M)	Ours	28.562	25.622	26.915	32.741
CDNSR(S)	Without KD	28.471	25.571	26.873	32.688
CDNSR(S)	Pixel Loss	28.468	25.570	26.870	32.676
CDNSR(S)	Perceptual Loss	28.424	25.544	26.835	32.644
CDNSR(S)	CSD Loss	28.480	25.574	26.874	32.675
CDNSR(S)	Ours	28.493	25.583	26.884	32.692

[33] is also based on the idea of contrastive learning, but can't achieve the same performance as ours. The underlying reason is that, the cosine distance is more capable to measure the distances between samples in the feature space than the Manhattan distance (ℓ_1-norm). The results in Table VI further show that the contrastive learning based knowledge distillation can always help to improve the performance of CDNSR(S) on different backbones.

TABLE VI
EFFECT OF KNOWLEDGE DISTILLATION FOR CDNSR(S) ON DIFFERENT BACKBONES.

Module-SR	DIV2K	Tesk2K	Tesk4K	Tesk8K
FSRCNN (without KD)	28.47	25.57	26.87	32.67
FSRCNN	28.49	25.58	26.88	32.69
CARN (without KD)	29.10	25.94	27.33	33.14
CARN	29.13	25.94	27.34	33.15
SRResNet (without KD)	29.52	26.18	27.64	33.47
SRResNet	29.53	26.19	27.65	33.49

5) Effect of the Number of Negative Samples.: Finally, we explore the effect of the number S of negative samples. As shown in Table VII, we can observe that adding not too many negative samples can achieve better performance. Another problem is, the model training would cost require more memory footprint and training time with the increment of negative samples. Considering the performance-cost trade-off, we choose $S = 6$ in our all experiments.

V. CONCLUSION

In this paper, we explore the computation redundancy to improve inference efficiency of SR backbones. Specifically, we develop CDNSR, a classification-based dynamic network for efficient super-resolution. The basic idea is to categorize

TABLE VII
EFFECT OF THE NUMBER OF NEGATIVE SAMPLES (S) ON FSRCNN.

Module-SR	S	Test2K	Test4K	Test8K
CDNSR(S)	2	25.580	26.882	32.689
CDNSR(S)	4	25.584	26.885	32.692
CDNSR(S)	6	25.585	26.888	32.695
CDNSR(S)	8	25.583	26.885	32.690

image-patches into different classes, each of which corresponds to a SR network matched with the restoration difficulty. The proposed CDNSR is decoupled with the backbone architecture for SR, and can be generalized to any SR backbone. We have comprehensively verified the efficiency of CDNSR for accelerating popular SR backbones on typical datasets with superior performance. Besides, CDNSR is more flexible and capable than the previous classification-based SR solution ClassSR.

REFERENCES

[1] K. Zhang, L. V. Gool, and R. Timofte, "Deep unfolding network for image super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 3217–3226.

[2] L. Wang, Y. Wang, X. Dong, Q. Xu, J. Yang, W. An, and Y. Guo, "Unsupervised degradation representation learning for blind super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 581–10 590.

[3] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, "Meta-sr: A magnification-arbitrary network for super-resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1575–1584.

[4] L. Wang, Y. Wang, Z. Lin, J. Yang, W. An, and Y. Guo, "Learning a single network for scale-arbitrary super-resolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4801–4810.

[5] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

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

[7] 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.

[8] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 286–301.

[9] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 252–268.

[10] Z. Hui, X. Wang, and X. Gao, "Fast and accurate single image super-resolution via information distillation network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 723–731.

[11] M. Zhu, K. Han, E. Wu, Q. Zhang, Y. Nie, Z. Lan, and Y. Wang, "Dynamic resolution network," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[12] 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. 12 016–12 025.

[13] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[14] Y. Wang, S. Lin, Y. Qu, H. Wu, Z. Zhang, Y. Xie, and A. Yao, "Towards compact single image super-resolution via contrastive self-distillation," *arXiv preprint arXiv:2105.11683*, 2021.

[15] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision*. Springer, 2016, pp. 391–407.

[16] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[17] Z. Hui, X. Gao, Y. Yang, and X. Wang, "Lightweight image super-resolution with information multi-distillation network," in *Proceedings of the 27th acm international conference on multimedia*, 2019, pp. 2024–2032.

[18] H. Zhao, X. Kong, J. He, Y. Qiao, and C. Dong, "Efficient image super-resolution using pixel attention," in *European Conference on Computer Vision*. Springer, 2020, pp. 56–72.

[19] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. 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*, 2016, pp. 1874–1883.

[20] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[21] K. Xu, L. Rui, Y. Li, and L. Gu, "Feature normalized knowledge distillation for image classification," in *European Conference on Computer Vision*. Springer, 2020, pp. 664–680.

[22] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3967–3976.

[23] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, "Online knowledge distillation with diverse peers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3430–3437.

[24] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *European Conference on Computer Vision*. Springer, 2020, pp. 319–345.

[25] H. Wu, Y. Qu, S. Lin, J. Zhou, R. Qiao, Z. Zhang, Y. Xie, and L. Ma, "Contrastive learning for compact single image dehazing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 551–10 560.

[26] G. Xu, Z. Liu, X. Li, and C. C. Loy, "Knowledge distillation meets self-supervision," in *European Conference on Computer Vision*. Springer, 2020, pp. 588–604.

[27] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[28] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Conference Track Proceedings on 5th International Conference on Learning Representations, ICLR 2017*, 2017.

[29] X. Wang and Y. Li, "Harmonized dense knowledge distillation training for multi-exit architectures," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, 2021, pp. 10 218–10 226.

[30] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[31] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[32] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," in *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018, pp. 0–0.

[33] S. Wang, M. Lu, K. Chen, J. Liu, X. Li, M. Wu *et al.*, "Samplingaug: On the importance of patch sampling augmentation for single image super-resolution," *arXiv preprint arXiv:2111.15185*, 2021.

[34] 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.

[35] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proceedings of the IEEE confer-*

ence on computer vision and pattern recognition workshops, 2017, pp. 126–135.

[36] S. Gu, A. Lugmayr, M. Danelljan, M. Fritsche, J. Lamour, and R. Timofte, "Div8k: Diverse 8k resolution image dataset," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 2019, pp. 3512–3516.