NTIRE 2023 Challenge on Efficient Super-Resolution: Methods and Results

Yawei Li* Yulun Zhang* Radu Timofte* Luc Van Gool* Lei Yu Youwei Li
Xinpeng Li Ting Jiang Qi Wu Mingyan Han Wenjie Lin Chengzhi Jiang
Jinting Luo Haoqiang Fan Shuaicheng Liu Yucong Wang Minjie Cai
Mingxi Li Yuhang Zhang Xian-jun Fan Yankai Sheng Yanyu Mao
Nihao Zhang Qian Wang Mingjun Zheng Long Sun Jinshan Pan
Jiangxin Dong Jinhui Tang Zhongbao Yang Yan Wang Erlin Pan Qixuan Cai
Xinan Dai Magauiya Zhussip Nikolay Kalyazin Dmitry Vyal Xueyi Zou
Youliang Yan Heaseo Chung Jin Zhang Gaocheng Yu Feng Zhang
Hongbin Wang Bohao Liao Zhibo Du Yu-liang Wu Gege Shi Long Peng
Yang Wang Yang Cao Zhengjun Zha Zhi-Kai Huang Yi-Chung Chen
Yuan-Chun Chiang Hao-Hsiang Yang Wei-Ting Chen Hua-En Chang
I-Hsiang Chen Chia-Hsuan Hsieh Sy-Yen Kuo Xin Liu Qian Wang Jiahao Pan
Hongyuan Yu Weichen Yu Lin Ge Jiahua Dong Yajun Zou Zhuoyuan Wu
Binnan Han Xiaolin Zhang Heng Zhang Xuanwu Yin Kunlong Zuo
Weijian Deng Hongjie Yuan Zengtong Lu Mingyu Ouyang Wenzhuo Ma
Nian Liu Hanyou Zheng Yuantong Zhang Junxi Zhang Zhenzhong Chen
Garas Gendy Nabil Sabor Jingchao Hou Guanghui He Yurui Zhu Xi Wang
Xueyang Fu Zheng-Jun Zha Daheng Yin Mengyang Liu Baijun Chen Ao Li
Lei Luo Kangjun Jin Ce Zhu Xiaoming Zhang Chengxing Xie Linze Li
Haiteng Meng Tianlin Zhang Tianrui Li Xiaole Zhao Zhao Zhang Baiang Li
Huan Zheng Suiyi Zhao Yangcheng Gao Jiahuan Ren Kang Hu Jingpeng Shi
Zhijian Wu Dingjiang Huang Jinchen Zhu Hui Li Qianru Xv Tianle Liu
Shizhuang Weng Gang Wu Junpeng Jiang Xianming Liu Junjun Jiang
Mingjian Zhang Shizhuang Weng Jing Hu Chengxu Wu Qinrui Fan
Chengming Feng Ziwei Luo Shu Hu Siwei Lyu Xi Wu Xin Wang

Abstract

This paper reviews the NTIRE 2023 challenge on efficient single-image super-resolution with a focus on the proposed solutions and results. The aim of this challenge is to devise a network that reduces one or several aspects such as runtime, parameters, FLOPs, activations, memory footprint, and depth of RFDN while at least maintaining the PSNR of 29.00dB on DIV2K validation datasets. The challenge had 272 registered participants, and 35 teams made valid submissions. They gauge the state-of-the-art for efficient single-image super-resolution.

1. Introduction

Single image super-resolution (SR) focuses on reconstructing a high-resolution (HR) image from a single low-resolution (LR) image that has undergone a specific degradation process [25, 54, 55, 65, 110]. In image SR, it is assumed that the LR image results from two major degradation processes including blurring and down-sampling. For classical image SR, bicubic down-sampling is the most commonly used degradation model. This classical standard degradation model allows for direct comparisons between

* Y. Li (yawei.li@vision.ee.ethz.ch, Computer Vision Lab, ETH Zurich), Y. Zhang, R. Timofte, and L. Van Gool were the challenge organizers, while the other authors participated in the challenge. Each team described their own method in the report.
Appendix A contains the authors' teams and affiliations.
NTIRE 2023 webpage: https://cvlai.net/ntire/2023/.
Code: https://github.com/ofsoundof/NTIRE2023_ESR.

different image SR methods and serves as a testbed to validate the advantages of newly proposed SR methods.

State-of-the-art deep neural networks for image SR usually comes with parameter overparameterization, intensive computation, high latency *etc.* And that makes it difficult to deploy those models on mobile devices for real-time SR. To solve the problem, there is a vast array of research that aims to improve the efficiency of deep neural networks through network pruning [57, 71], low-rank filter decomposition [56, 108], network quantization, neural architecture search [58, 115, 116], and knowledge distillation [36, 86]. Some of these network compression techniques have been successfully applied to image SR.

The efficiency of a deep neural network can be measured using different metrics, including runtime, number of parameters, computational complexity (FLOPs), activations, and memory consumption. These metrics affect the deployment of deep neural networks in various ways. Among them, runtime is the most direct indicator of a network's efficiency and is used as the primary evaluation metric. Increased computational complexity is associated with higher energy consumption, which could shorten the battery life of mobile devices. Memory efficiency is also an important metric on edge devices. Furthermore, the number of parameters is related to AI chip design, with more parameters resulting in larger chip areas and increased costs for the designed AI devices.

In collaboration with the 2023 New Trends in Image Restoration and Enhancement (NTIRE 2023) workshop, we organize the challenge on efficient super-resolution. The challenge's goal is to super-resolve an LR image with a magnification factor of $\times 4$ using a network that reduces aspects such as runtime, parameters, FLOPs, activations, and memory consumption of RFDN [66] while maintaining a PSNR of at least 29.00dB on the DIV2K validation set. This challenge aims to discover advanced and innovative solutions for efficient SR, benchmark their efficiency, and identify general trends for the design of efficient SR networks.

This challenge is one of the NTIRE 2023 Workshop series of challenges on: night photography rendering [79], HR depth from images of specular and transparent surfaces [101], image denoising [61], video colorization [45], shadow removal [87], quality assessment of video enhancement [69], stereo super-resolution [89], light field image super-resolution [92], image super-resolution ($\times 4$) [111], 360° omnidirectional image and video super-resolution [5], lens-to-lens bokeh effect transformation [16], real-time 4K super-resolution [17], HR nonhomogenous dehazing [2], efficient super-resolution (this challenge).

https://cvlai.net/ntire/2023/

2. NTIRE 2023 Efficient Super-Resolution Challenge

The goals of this challenge include: (1) promoting research in the area of efficient super-resolution, (2) facilitating comparisons between the efficiency of various methods, and (3) providing a platform for academic and industrial participants to engage, discuss, and potentially establish collaborations. This section delves into the specifics of the challenge.

2.1. Dataset

The DIV2K [1] dataset and LSDIR [59] dataset are utilized for this challenge. DIV2K dataset consists of 1,000 diverse 2K resolution RGB images, which are split into a training set of 800 images, a validation set of 100 images, and a test set of 100 images. LSDIR dataset contains 86,991 high-resolution high-quality images, which are split into a training set of 84,991 images, a validation set of 1,000 images, and a test set of 1,000 images. In this challenge, the corresponding LR DIV2K images are generated by bicubic downsampling with a down-scaling factor of 4x. The training images from DIV2K and LSDIR are provided to the participants of the challenge. During the validation phase, the 100 images from DIV2K validation set were made available to participants. During test phase, 100 images from DIV2K test set and another 100 images from LSDIR test set are used. Throughout the entire challenge, the testing HR images remained hidden from participants.

2.2. RFDN Baseline Model

The Residual Feature Distillation Network (RFDN) [66] serves as the baseline model in this challenge. The aim is to improve its efficiency in terms of runtime, number of parameters, FLOPs, number of activations, and GPU memory consumption while maintaining a PSNR performance of 29.00dB on the validation set. RFDN is composed of four components: an initial feature extraction convolution, multiple stacked Residual Feature Distillation Blocks (RFDBs), a feature fusion layer, and a final reconstruction block. Specifically, the initial feature extraction is carried out by a 3×3 convolution that generates coarse features from the input LR image. The second part of RFDN consists of four RFDBs, stacked in a chain-like manner, to progressively refine the extracted features. After gradual refinement by the RFDBs, all intermediate features are combined using a 1×1 convolution layer. An additional 3×3 convolution layer is then utilized to smooth the aggregated features. Finally, the super-resolved images are generated by pixel shuffle operation.

The baseline RFDN is provided by the winner of the AIM 2020 Challenge on Efficient Super-Resolution [104]. The quantitative performance and efficiency metrics of

RFDN are given in Tab. 1 and summarized as follows. (1) The number of parameters is 0.433M. (2) The average PSNRs on validation (DIV2K 100 validation images) and testing (DIV2K 100 test images and LSDIR 100 test images) sets of this challenge are 29.04dB and 27.11dB, respectively. (3) The runtime averaged on the validation and test set with PyTorch 1.11.0, CUDA Toolkit 10.2, cuDNN 7.6.2 and a single Titan Xp GPU is 29.67 ms. (4) The number of FLOPs for an input of size 256×256 is 27.10G. (5) The number of activations (*i.e.* the number of elements in all convolutional layer outputs) for an input of size 256×256 is 112.03M. (5) The maximum GPU memory consumption during the inference on the DIV2K validation set is 780.13M. (6) The number of convolutional layers is 64.

2.3. Tracks and Competition

The aim of this challenge is to devise a network that reduces one or several aspects such as runtime, parameters, FLOPs, activations and memory consumption while at least maintaining the PSNR of 29.00dB on the validation set.

Ranking statistic Similar to the previous year [60], to determine the ranking in the case of multiple metrics, the individual rankings of each metric are added up to form a ranking statistic of the metrics.

Challenge phases *(1) Development and validation phase*: Participants were given access to 800 LR/HR training image pairs and 100 LR/HR validation image pairs from the DIV2K dataset. Additional 84,991 LR/HR training image pairs from the LSDIR dataset are also provided to the participants. The RFDN model, pre-trained parameters, and validation demo script are available on GitHub https://github.com/ofsoundof/NTIRE2022_ESR, allowing participants to benchmark their models' runtime on their systems. Participants could upload their HR validation results to the evaluation server to calculate the PSNR of the super-resolved image produced by their models and receive immediate feedback. The number of parameters and runtime were computed by the participants. *(2) Testing phase*: In the final test phase, participants were granted access to 100 LR testing images from DIV2K and 100 LR testing images from LSDIR, while the HR ground-truth images remained hidden. Participants submitted their super-resolved results to the Codalab evaluation server and emailed the code and factsheet to the organizers. The organizers verified and ran the provided code to obtain the final results, which were then shared with participants at the end of the challenge.

Evaluation protocol Quantitative evaluation metrics included validation and testing PSNRs, runtime, number of parameters, FLOPs, activations, and maximum GPU memory consumption during inference. PSNR was measured by discarding a 4-pixel boundary around the images. The average runtime during inference on the 100 LR validation images and the 200 LR testing images was computed. The best runtime among three consecutive trials was selected as the final result. The average runtime on the validation and testing sets served as the final runtime indicator. Maximum GPU memory consumption was recorded during inference. FLOPs and activations were evaluated on an input image of size 256x256. Among these metrics, runtime was considered the most important. Participants were required to maintain a PSNR of 29.00dB on the validation set during the challenge. The constraint on the testing set helped prevent overfitting on the validation set. A code example for calculating these metrics is available at https://github.com/ofsoundof/NTIRE2023_ESR. The code of the submitted solutions and the pre-trained weights are also available in this repository.

3. Challenge Results

The final test results and rankings are presented in Tab. 1. To maintain the fairness of the competition, any solutions with test PSNR lower than 26.95dB are not included in the rankings. The table also includes the baseline method RFDN [66] for comparison. In Sec.4, the methods evaluated in Tab. 1 are briefly explained, while the team members are listed in Appendix A. The performance of different methods is compared from four different perspectives including two running metrics (runtime and GPU memory footprint) and two combined metrics (model complexity and overall performance). Additionally, to further foster fair competition of this challenge, the image reconstruction quality in terms of test PSNR are compared among teams with test runtime less than 30ms (real-time inference [17]). The observations that can be drawn from Tab. 1 are as follows.

Runtime. First, the runtime is the important evaluation metric in this challenge. The solution proposed by MegSR has the smallest runtime in this efficient SR challenge. The Zapdos and DFCDN win second and third place, respectively. The runtime of the first three solutions averaged on the validation and test set is below 18 ms. The first 15 teams proposed a solution with average runtime lower than 30 ms. The proposed solutions continue to improve the efficiency of image SR networks. The difference between the runtime of the first three teams are quite small, indicating the competitiveness of the challenge. In addition, DFCDN team achieves the highest PSNR on the test set among the first three teams.

GPU memory footprint. The memory footprint is quite an important running metric for efficient models. Thus, the models that optimize towards this direction are also singled out. The two solutions proposed by NoahTerminalCV consume the least GPU memory. The second runner-up goes to

Table 1. Results of NTIRE 2023 Efficient SR Challenge. The performance of the solutions are compared thoroughly from several perspective including the runtime, the nubmer of parameters, FLOPs, the number of activations, and GPU memory footprint. The underscript numbers associated with each metric denote the ranking of the solution in terms of that metric. "Val. Time" is the runtime averaged on DIV2K [1] validation set. "Test Time" is the runtime averaged on a test set with 100 images from DIV2K [1] and LSDIR [59] test set, respectively. "Ave. Time" is averaged on the validation and test datasets. "#Params" is the total number of parameters of a model. "FLOPs" denotes the floating point operations. "#Acts" denotes the number of elements of all convolutional layer outputs. "GPU Mem." represents maximum GPU memory footprint measured according to the PyTorch function `torch.cuda.max_memory_allocated()` during the inference on DIV2K validation set. "#Conv" represents the number of convolutional layers. "FLOPs" and "#Acts" are tested on an LR image of size 256×256. "Model Comp." is a combined metric based on the number of parameters and FLOPs that reflects theoretical model complexity. "Overall" denotes the metric that combines all the five evaluation metrics including runtime, number of parameters, FLOPs, number of activations, and GPU memory footprint. **This is not a challenge for PSNR improvement. The "validation/testing PSNR" and "#Conv" are not ranked**.

Team	Time [ms]			PSNR [dB]		#Params	FLOPs	#Acts	GPU Mem.	Model	Overall	#Conv
	Ave.	Val.	Test	Val.	Test	[M]	[G]	[M]	[M]	Comp.		
MegSR	$18.30_{(1)}$	21.26	15.33	29.04	26.95	$0.243_{(12)}$	$14.90_{(11)}$	$72.97_{(6)}$	$495.91_{(19)}$	$23_{(11)}$	$49_{(2)}$	39
Zapdos	$18.59_{(2)}$	21.69	15.48	28.96	27.03	$0.352_{(25)}$	$21.97_{(25)}$	$63.01_{(2)}$	$420.50_{(13)}$	$50_{(25)}$	$67_{(10)}$	26
DFCDN	$18.71_{(3)}$	21.91	15.51	29.00	27.08	$0.245_{(13)}$	$15.49_{(14)}$	$82.76_{(13)}$	$376.99_{(12)}$	$27_{(14)}$	$55_{(4)}$	39
KaiBai Group	$20.49_{(4)}$	23.94	17.05	28.95	27.01	$0.272_{(17)}$	$16.76_{(17)}$	$65.10_{(3)}$	$296.45_{(7)}$	$34_{(17)}$	$48_{(1)}$	35
R.I.P. ShopeeVideo	$20.65_{(5)}$	24.34	16.96	28.97	27.04	$0.255_{(15)}$	$16.16_{(16)}$	$74.97_{(7)}$	$439.60_{(14)}$	$31_{(15)}$	$57_{(5)}$	35
Antins_cv	$20.92_{(6)}$	24.45	17.39	29.00	26.95	$0.315_{(24)}$	$20.07_{(24)}$	$70.82_{(5)}$	$488.61_{(17)}$	$48_{(24)}$	$76_{(14)}$	29
Young	$22.09_{(7)}$	25.86	18.33	28.97	27.00	$0.543_{(30)}$	$33.38_{(30)}$	$61.87_{(1)}$	$293.05_{(6)}$	$60_{(30)}$	$74_{(13)}$	23
NTU607_ESR	$22.71_{(8)}$	26.73	18.68	29.00	27.07	$0.281_{(19)}$	$17.31_{(19)}$	$76.11_{(9)}$	$364.24_{(11)}$	$38_{(19)}$	$66_{(8)}$	39
CMVG	$24.42_{(9)}$	28.51	20.33	29.01	27.08	$0.307_{(21)}$	$18.98_{(21)}$	$81.55_{(11)}$	$454.51_{(15)}$	$42_{(21)}$	$77_{(15)}$	41
Touch_Fish	$25.61_{(10)}$	30.09	21.12	29.00	27.09	$0.415_{(27)}$	$27.16_{(27)}$	$75.50_{(8)}$	$769.56_{(27)}$	$54_{(27)}$	$99_{(26)}$	20
CUC_SR	$25.97_{(11)}$	30.58	21.37	28.99	27.05	$0.402_{(26)}$	$25.23_{(26)}$	$81.88_{(12)}$	$344.51_{(9)}$	$52_{(26)}$	$84_{(20)}$	39
SeaOuter	$26.26_{(12)}$	30.93	21.59	28.95	27.05	$0.285_{(20)}$	$18.63_{(20)}$	$80.48_{(10)}$	$218.97_{(3)}$	$40_{(20)}$	$65_{(7)}$	44
NoahTerminalCV B	$27.83_{(13)}$	32.73	22.94	28.96	27.03	$0.209_{(10)}$	$13.34_{(10)}$	$118.71_{(18)}$	$188.21_{(1)}$	$20_{(10)}$	$52_{(3)}$	49
NJUST_R	$28.63_{(14)}$	33.59	23.66	28.99	27.07	$0.237_{(11)}$	$15.40_{(13)}$	$86.11_{(14)}$	$303.06_{(8)}$	$24_{(12)}$	$60_{(6)}$	58
NoahTerminalCV A	$28.71_{(15)}$	33.74	23.69	28.99	27.06	$0.310_{(23)}$	$19.99_{(23)}$	$68.38_{(4)}$	$188.60_{(2)}$	$46_{(23)}$	$67_{(9)}$	25
Sissie_Lab	$30.34_{(16)}$	34.62	26.07	29.00	27.00	$0.461_{(28)}$	$28.85_{(28)}$	$107.07_{(16)}$	$628.94_{(25)}$	$56_{(28)}$	$113_{(29)}$	48
GarasSjtu	$32.30_{(17)}$	37.99	26.62	28.91	26.99	$0.275_{(18)}$	$16.85_{(18)}$	$97.87_{(15)}$	$556.22_{(22)}$	$36_{(18)}$	$90_{(22)}$	43
USTC_ESR	$34.16_{(18)}$	40.11	28.20	29.03	27.09	$0.503_{(29)}$	$31.56_{(29)}$	$112.57_{(17)}$	$489.33_{(18)}$	$58_{(29)}$	$111_{(28)}$	48
SEU_CNII	$40.84_{(19)}$	48.35	33.33	28.99	27.08	$0.616_{(31)}$	$38.63_{(31)}$	$133.57_{(19)}$	$944.91_{(29)}$	$62_{(31)}$	$129_{(31)}$	64
AVC2_CMHI_SR	$43.46_{(20)}$	51.30	35.61	29.01	27.06	$0.262_{(16)}$	$15.52_{(15)}$	$154.19_{(20)}$	$821.45_{(28)}$	$31_{(16)}$	$99_{(25)}$	84
NJUST_M	$68.11_{(21)}$	79.54	56.68	28.96	27.05	$0.104_{(3)}$	$6.56_{(3)}$	$199.35_{(23)}$	$503.49_{(20)}$	$6_{(3)}$	$70_{(12)}$	66
TelunXupt	$75.89_{(22)}$	88.10	63.68	29.00	27.09	$0.095_{(1)}$	$5.58_{(1)}$	$220.88_{(25)}$	$517.14_{(21)}$	$2_{(1)}$	$70_{(11)}$	317
Set5 Baby	$99.79_{(23)}$	117.33	82.25	29.01	27.08	$0.129_{(6)}$	$8.29_{(5)}$	$202.70_{(24)}$	$652.41_{(26)}$	$11_{(5)}$	$84_{(19)}$	86
NJUST_E	$106.61_{(24)}$	125.02	88.20	28.97	27.04	$0.099_{(2)}$	$6.02_{(2)}$	$242.96_{(26)}$	$606.38_{(24)}$	$4_{(2)}$	$78_{(16)}$	66
LVGroup_HFUT	$112.68_{(25)}$	132.10	93.26	28.98	27.05	$3.426_{(32)}$	$224.19_{(32)}$	$335.28_{(30)}$	$590.58_{(23)}$	$64_{(32)}$	$142_{(32)}$	94
FRL Team 4	$124.13_{(26)}$	145.18	103.07	28.95	27.02	$0.173_{(7)}$	$10.60_{(7)}$	$187.32_{(22)}$	$1266.92_{(31)}$	$14_{(7)}$	$93_{(23)}$	198
Dase-IDEALab	$130.73_{(27)}$	153.30	108.17	29.00	27.07	$0.118_{(5)}$	$9.06_{(6)}$	$332.39_{(29)}$	$1114.77_{(30)}$	$11_{(6)}$	$97_{(24)}$	122
FRL Team 1	$186.02_{(28)}$	218.82	153.23	29.01	27.03	$0.200_{(9)}$	$12.76_{(9)}$	$243.20_{(27)}$	$265.25_{(5)}$	$18_{(9)}$	$78_{(18)}$	100
FRL Team 0	$196.64_{(29)}$	230.14	163.14	29.01	26.98	$0.115_{(4)}$	$7.38_{(4)}$	$170.26_{(21)}$	$2028.66_{(32)}$	$8_{(4)}$	$90_{(21)}$	58
FRL Team 3	$201.34_{(30)}$	237.73	164.94	29.00	27.09	$0.179_{(8)}$	$11.54_{(8)}$	$285.41_{(28)}$	$262.67_{(4)}$	$16_{(8)}$	$78_{(17)}$	112
AIIA-SR	$224.45_{(31)}$	264.99	183.91	29.00	27.07	$0.307_{(22)}$	$19.53_{(22)}$	$355.47_{(31)}$	$482.70_{(16)}$	$44_{(22)}$	$122_{(30)}$	89
FRL Team 2	$282.42_{(32)}$	331.55	233.29	29.02	27.02	$0.245_{(14)}$	$15.37_{(12)}$	$422.90_{(32)}$	$355.94_{(10)}$	$26_{(13)}$	$100_{(27)}$	158
The following methods are not ranked since their validation/testing PSNR are not on par with the baseline.												
CUIT_SRLab	175.66	206.24	145.08	27.07	25.44	0.183	11.72	256.90	356.45			66
Loading2	2535.11	2977.98	2092.25	27.07	25.44	11.900	500.56	185.60	2691.86			12
Alpha	26.87	31.16	22.57	28.81	26.88	0.198	11.23	86.18	730.99			56
RFDN baseline	35.54	42.41	28.66	29.04	27.11	0.433	27.10	112.03	788.13			64

the SeaOuter team. The GPU memory footprint of all three solutions is below 220 MB.

Model Complexity. The number of parameters and FLOPs are among the most important metrics to indicate the model complexity. In this combined metric, TelunXupt proposes a method with both the smallest number of parameters and FLOPs. Thus, TelunXupt is the winner under this combined metric. The 1st and 2nd runner-ups are the NJUST_E and NJUST_M, respectively.

Overall evaluation. Finally, the performance with respect to the overall metric that combines all the five evaluation metrics including runtime, number of parameters, FLOPs, number of activations, and GPU memory footprint is also reported. Under this metric, the KaiBai Group wins

first place. The first and second runner-up teams are MegSR and NoahTerminalCV B, respectively.

Test PSNR. Since this challenge is intended for efficient SR, the test PSNR is only compared among teams with test runtime less than 30ms. This is the minimum requirement for real-time inference [17]. In total, 18 teams meet this requirement. Among the 18 teams, Team USTC_ESR and Team Touch_Fish achieve the best PSNR of 27.09 dB while the test runtime of Touch_Fish is reduced by 7.08ms. Team DFCDN and Team CMVG achieve the second best PSNR of 27.08 dB. Compared with CMVG, the solution proposed by DFCDN is 4.82 ms faster.

3.1. Fairness

To ensure the fairness of the efficient SR challenge, several rules were established, mainly concerning the dataset used for training the network. First, training with additional external datasets, such as Flickr2K, was allowed. Second, training with the additional DIV2K validation set, including either HR or LR images, was not permitted, as the validation set was used to assess the overall performance and generalizability of the proposed network. Third, training with DIV2K test LR images was prohibited. Lastly, using advanced data augmentation strategies during training was considered a fair approach.

3.2. Conclusions

Several conclusions can be drawn from the analysis of different solutions as follows. Firstly, the proposed methods improve the state-of-the-art for efficient SR. Secondly, network compression methods begin to play an important role. Noticeably, knowledge distillation is used in the winner and first runner-up solutions in terms of runtime. Network pruning is used by the winning team MegSR to further improve the efficiency of the network. Thirdly, the adoption of large-scale dataset [59] for pre-training improves the accuracy of the network. Fourthly, for most of the methods, the training of the network proceeds in several phases with increased patch size and reduced learning rate. And finally, by jointly considering runtime, number of parameters, FLOPs, number of activations, and memory footprint, it is possible to design a balanced model that optimizes more than one evaluation metric.

4. Challenge Methods and Teams

4.1. MegSR

General method description. The MegSR team proposed an efficiency distillation and iterative pruning SR network named DIPNet [98]. As shown in Fig. 1, the method consists of three stages. In the first stage, this team trains a large teacher network with Hybrid Attention Transformer [9] backbone, denoted as $\mathcal{T}$. It is worth noting that they do not directly use the high-resolution image I_{HR} provided in the training dataset for $\times 4$ super-resolution training. Inspired by HGGT [6], this team first trains a network for $\times 1$ super-resolution. The HR image I_{HR} is then utilized as input for $\times 1$ super-resolution, yielding an enhanced HR output I_{enh}, and the low resolution image I_{LR} and the enhanced HR I_{enh} is used for $\times 4$ super-resolution training through minimize the following loss:

$$L_{\mathcal{T}} = ||\mathcal{T}(I_{LR}) - I_{enh}||_1 \tag{1}$$

After training the teacher network, this team performs multi-level distillation on their proposed student network, denoted as $\mathcal{S}$. Their student network is modified based on RLFN [50]. They expand the RLFB in RLFN to the structure RRFB shown in Fig. 1(b). Once the student network training converges, it can be restored to the RLFB structure by the reparameterization technique. During distillation, this team uses the feature maps extracted from four different depths of $\mathcal{T}$ to supervise the learning of each of the four blocks in $\mathcal{S}$. Specifically, they minimize the following losses:

$$L_{feat} = \lambda_i \sum_{i=1}^{4} ||F_i^{\mathcal{T}} - \psi(F_i^{\mathcal{S}})||_1, \tag{2}$$

where $F_i^{\mathcal{S}}$ represents the feature map of the output of the i-th block of $\mathcal{S}$, while $F^{\mathcal{T}}$ represents the feature of the output of some RHAGs of $\mathcal{T}$, ψ represents the operation of using a 1×1 convolution to expand the feature channels of $\mathcal{S}$ to the number of feature channels of $\mathcal{T}$, and λ_i is a weight for controlling the importance of the supervision from each depth level. This team also uses the outputs of $\mathcal{T}$ as pseudo-gt and enhanced gt to further supervise the learning of $\mathcal{S}$. Specifically, they compute the following losses:

$$L_{out} = ||\mathcal{T}(I_{LR}) - \mathcal{S}(I_{LR})||_1 + ||\mathcal{S}(I_{LR}) - I_{enh}||_1, \tag{3}$$

After distillation, the team employs a progressive learning strategy to finetune $\mathcal{S}$. They gradually increase the size of the input patch while using L_2 loss for supervised training until the model fully converges. Then, they reparameterize the model to further compress its size.

In the third stage, they iteratively pruned the reparameterized student network $\mathcal{S}$:

$$\mathcal{S}_p^i = \varphi(\Phi(\mathcal{S}_p^{i-1}; r)), \tag{4}$$

where Φ is the pruning operation, r is the pruning rate, φ means the finetuning operation, $\mathcal{S}_p^i$ means the network after the i-th pruning. Inspired by AGP [114], L_2 filter pruning is used in our iterative pruning method for model training. We stop pruning until the network cannot make effective predictions, and use the network obtained from the last effective pruning as the final network.

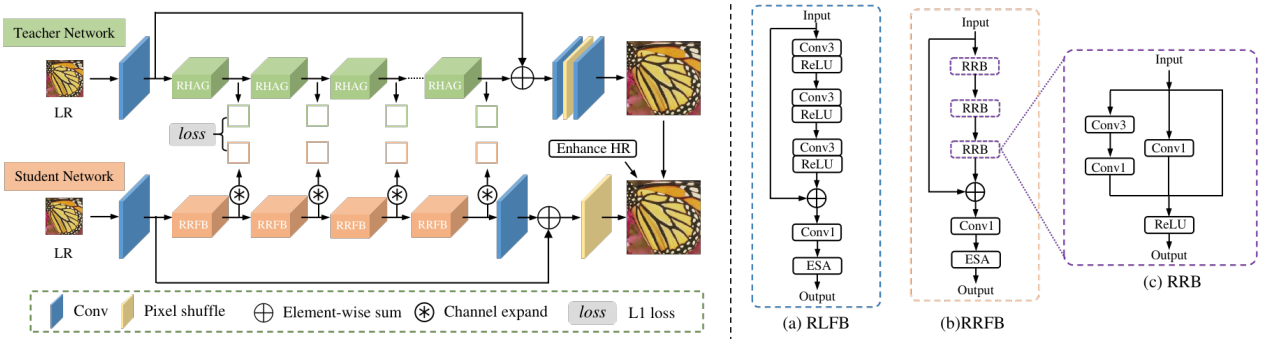


Figure 1. *Team MegSR:* The pipeline of DIPNet. (a) Structure of RLFB. (b) The structure of RRFB.

Training description. All training by this team is done on NVIDIA 2080Ti using only the training set of the DIV2K [1] dataset. During the training phase, they use random flip and rotation augmentation, and they chose Adam as the optimizer. When training the teacher net and distilling the student network, they set an initial learning rate of 1×10^{-4}, halved the learning rate every 100,000 iterations, and then used L_1 loss for supervision. When using the progressive learning strategy to finetune the distilled network, they use an initial learning rate of 2×10^{-5}, which is halved every 20,000 iterations. In this process, the training patch size is progressively increased to improve the performance, which is selected from $[64, 128, 256, 384]$. In the iterative pruning process, the ratio of each pruning is 0.05, and it is repeated three times in total. After each pruning, I_{LR} and I_{enh} are used for finetune, and 384×384 patches are used as input during finetuning.

4.2. Zapdos

General method description. The Zapdos team proposed Single Residual Network (SRN) for efficient image SR [91]. The proposed network has four Non-Residual Blocks (NRBs), in which the number of feature channels is set to 64 while the channel number of ESA [51] is set to 16. This network is used as a student network. The difference between the original EDSR [65] and the teacher model can be seen in Fig. 2 which also includes the student network. Considering the limited resources, the original setting of channel size (256) and number of residual blocks (32) is not adopted. Instead, the number of feature channels is set to 128, and the number of residual blocks is set to 20. The residual scaling factor is set to 1. More information can be seen from Fig. 3 and Fig. 4.

Training strategy. In total, two datasets are used including DIV2K [1] and LSDIR [59]. To train the models with images, the training dataset is augmented with geometric transforms: vertical/horizontal flips and 90-degree rotation in order to enhance the comprehensive ability of the model.

For the teacher model:

1. In the first stage, the model is trained from scratch. HR patches of size 192 × 192 are randomly cropped from HR images, and the mini-batch size is set to 16. The teacher model is trained by minimizing L1 loss function with Adam optimizer. The initial learning rate is set to 2×10^{-4}. The total number of epochs is 20000. (Only use DIV2K [1] dataset). The learning rate decay follows cosine annealing with T_{max} = total epochs, $\eta_{min} = 1 \times 10^{-7}$.

2. In the second stage, the model is initialized with the pre-trained weights. The initial learning rate is set to 1×10^{-4}. In this stage, we use LDSIR [59] dataset. The total number of epochs is 200. Other settings are the same as in the previous step.

3. At the last stage, the model is initialized with the pre-trained weights. HR patches of size 256×256 are randomly cropped from HR images. The initial learning rate is set to 2.5×10^{-5}. Now the teacher model is trained by minimizing L2 loss function with Adam optimizer. The total number of epochs is 50. Other settings are the same as in the previous step. After training, we freeze the parameters of the teacher model.

For the student model: We only use the DIV2K [1] dataset. We use main loss and distillation loss to train our student model. Details can be seen in Fig. 5.

1. In the first stage, the model is trained from scratch. HR patches of size 256×256 are randomly cropped from HR images, and the mini-batch size is set to 32. The student model is trained by minimizing L1 loss function with Adam optimizer. The initial learning rate is set to 2×10^{-4}. The total number of epochs is 80000. The learning rate decay is following cosine annealing with T_{max} = total epochs, $\eta_{min} = 1 \times 10^{-7}$.

2. In the second stage, the model is initialized with the pre-trained weights, and trained with the same settings as in the previous step.

3. At the last stage, the model is initialized with the pre-trained weights. HR patches of size 640×640 are randomly cropped from HR images, and the mini-batch size is set to

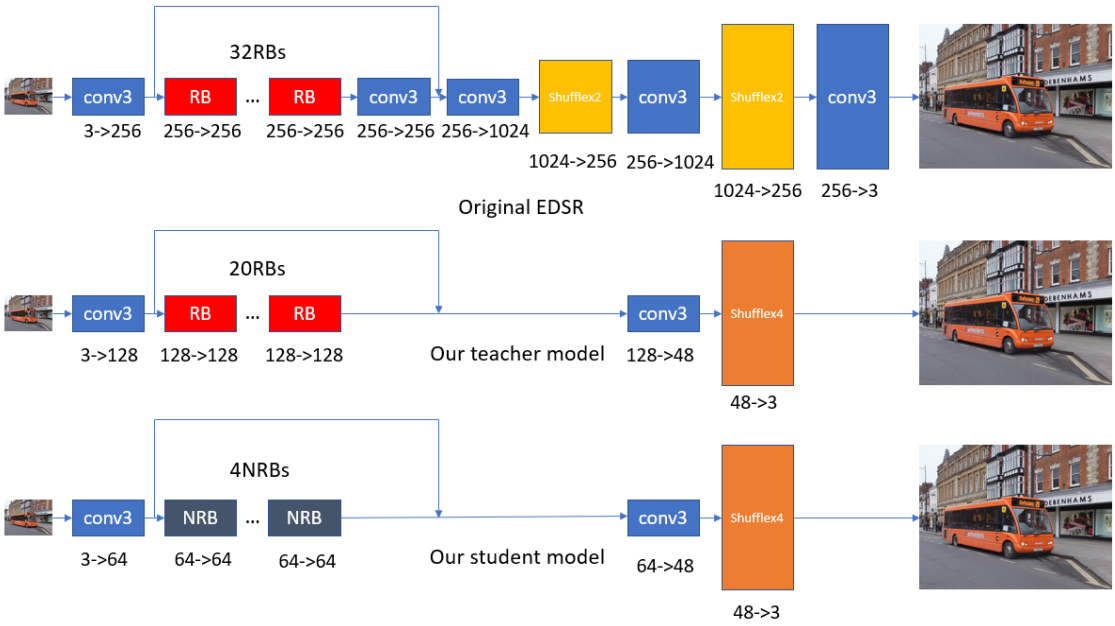


Figure 2. *Team Zapdos:* Original EDSR [65] and our models

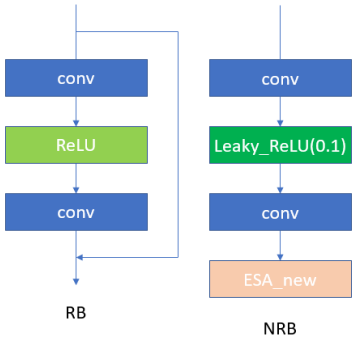


Figure 3. *Team Zapdos:* RB and NRB

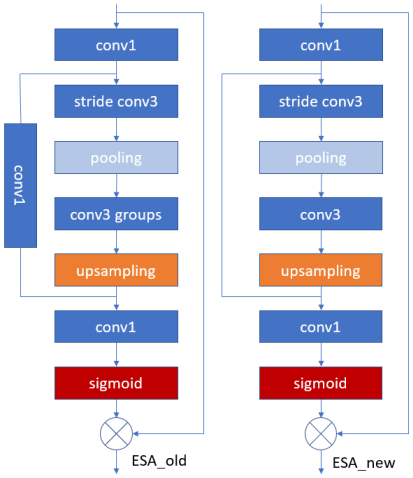


Figure 4. *Team Zapdos:* ESA [51] modify

32. The student model is trained by minimizing L2 loss function with Adam optimizer. The initial learning rate is set to 5×10^{-5}. The total number of epochs is 4000.

4.3. DFCDN

General method description. The overall architecture of Team DFCDN is shown in Fig. 6. The proposed network consists of four deep feature complement and distillation blocks (DFCDB). Inspired by [34], the input feature map is split equally along the channel dimension in each block. Then several convolutional layers process one of the split feature maps to generate complement features. The input features and complementary features are concatenated to avoid loss of input information and distilled by a conv-1 layer. Besides, the output feature map of DFCDB is further enhanced by ESA layer [51].

Online Convolutional Re-parameterization. Re-parameterization [106] has improved the performance of image restoration models without introducing any inference cost. However, the training cost is large because of complicated training-time blocks. To reduce the large extra training cost, we apply online convolutional re-parameterization [40] by converting the complex blocks into one single convolutional layer. The architecture of RepConv is shown in Fig. 6c, which can be converted to a

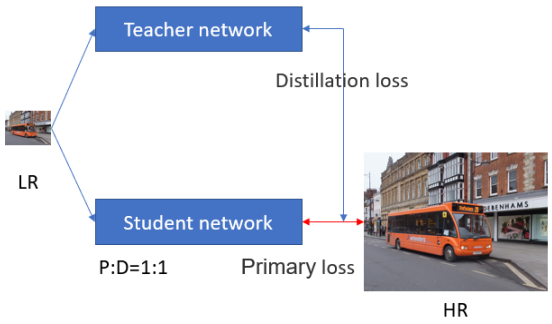


Figure 5. *Team Zapdos:* Loss

3×3 convolution during training.

Training Details. The proposed DFCDN has four DFCDBs. The number of features is set to 60 and the number of ESA channels is set to 16. DIV2K [1], Flickr2K [65] and LSDIR [59] datasets are used. The training details are as follows:

1. The model is first trained from scratch with 256×256 patches randomly cropped from HR images from DIV2K, Flickr2K and LSDIR datasets. The mini-batch size is set to 64. The L1 loss is minimized with Adam optimizer. The initial learning rate is set to 5×10^{-4} with a cosine annealing schedule. The total number of epochs is 1000.

2. In the second stage, the model is initialized with the pre-trained weights of Stage 1. The HR patch size is set to 640. The model is trained with the same settings as in the previous step.

3. In the third stage, the model is initialized with the pre-trained weights of Stage 2. The MSE loss is used for fine-tuning with 640×640 HR patches and a learning rate of 1×10^{-5} for 100 epochs.

4.4. TelunXupt

Method Details. The TelunXupt team proposed a multi-level dispersion residual network (MDRN) [73]. The lightweight distillation framework, as shown in Fig. 7, can improve the performance of deep feature extraction for SR, which has been verified in IMDN [42], RFDN [66] and BSRN [62]. The enhanced attention distillation block (EADB) is proposed to enhance the features with more efficient and powerful space and channel attention modules, as the based block of MDRN in Fig. 8. As for the space attention module, multi-level dispersion spatial attention (MDSA) is a substitute for enhanced spatial attention (ESA) [68]. In ESA, the dispersion branch (denoted as the branch-A) carries out the single large-size spatial compression and attention weight dispersion process *i.e.* , strided convolution and pooling operation, interpolation operation), and the refinement branch (denoted as the branch-B) uses one 1×1 convolution to map high-resolution features to the end. As shown in Fig. 9 (a), the dispersion branch (branch-A in ESA) is extended in a multi-level manner in MDSA. Except for the multi-level branches (D3, D5, and D7), MDSA further introduces the local variance (L-var) into the dispersion branch to better capture the area with rich structural information. To avoid introducing too many operations, local variance is only introduced into the dispersion branch D7. Besides, MDSA reduces the depth of Conv Groups in ESA to balance performance and model complexity. Inspired by RCAN [109], ECCA combines blueprint shallow residual block (BSRB) in BSRN with contrast-aware channel attention (CCA) [42] to form a residual structure as shown in Fig. 9 (b). Then, ECCA removes the residual connection in BSRB, and inserts the CCA module between the point-wise and depth-wise convolution layers.

Training strategy. The proposed MDRN has 8 EADBs, in which the number of feature channels is set to 28. The details of the training steps are as follows:

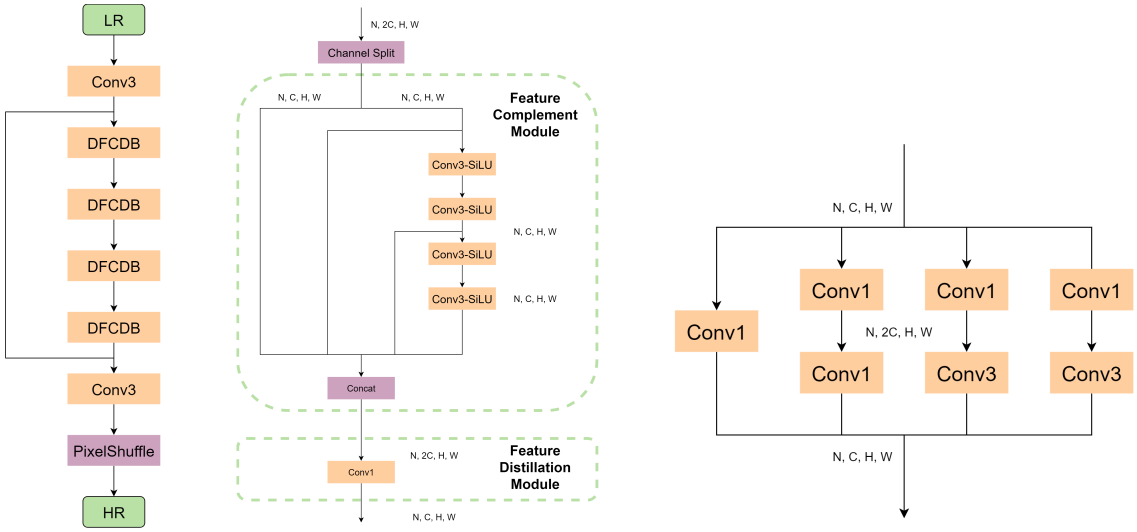
1. Pretraining on DIV2K [1]. HR patches of size 384×384 are randomly cropped from HR images, and the mini-batch size is set to 64. The model is trained by minimizing L1 loss function with Adam optimizer. The initial learning rate is set to 2×10^{-3} and halved at $\{100k, 500k, 800k, 900k, 950k\}$-iteration. The total number of iterations is 1000k.

2. Finetuning on 800 images of DIV2K and the first 10k images of LSDIR [59]. HR patch size and mini-batch size are set to 384×384 and 64, respectively. The model is fine-tuned by minimizing the Charbonnier loss function. The initial learning rate is set to 5×10^{-4} and halved at $\{100k, 500k, 800k, 900k, 950k\}$-iteration. The total number of iterations is 1000k.

3. Finetuning on 800 images of DIV2K and the first 10k images of LSDIR again. HR patch size and the mini-batch size are set to 480×480 and 64, respectively. The model is fine-tuned by minimizing L2 loss function. The initial learning rate is set to 2×10^{-4} and halved at $\{100k, 300k, 600k\}$-iteration. The total number of iterations is 650k.

4.5. NJUST_E

General method description. The NJUST_E proposes a lightweight multi-scale feature attention (MFA) for image super-resolution. The proposed MFA shown in Fig. 10, which consists of three stages including the shallow feature extraction, the deep feature extraction, and reconstruction.



(a) DFCDN (b) DFCDB (c) RepConv

Figure 6. *DFCDN Team:* The overall architecture of the proposed network. (a) Deep feature complement and distillation network (DFCDN). (b) Deep feature complement and distillation block (DFCDB). (c) RepConv.

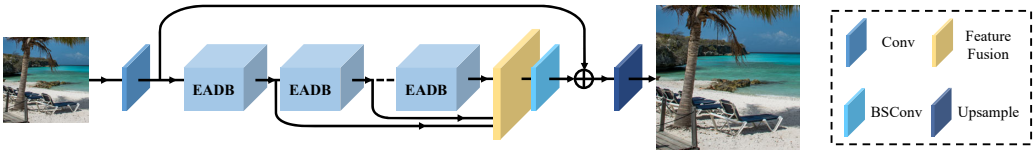
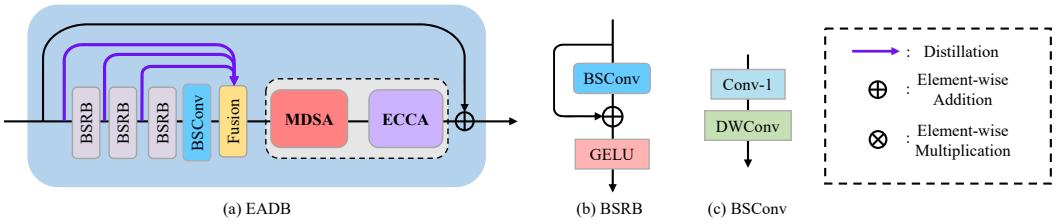
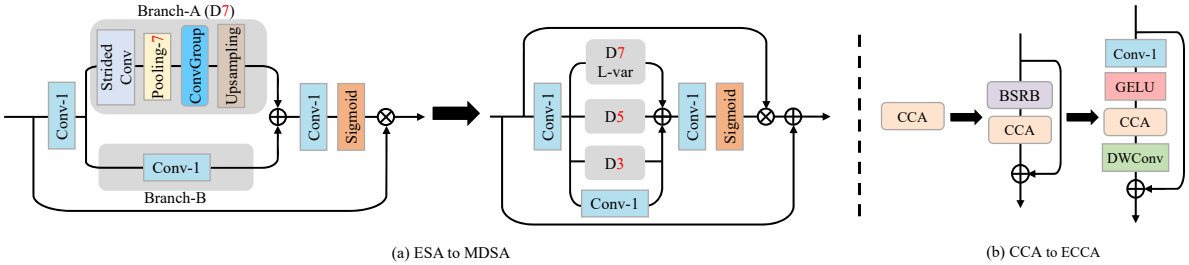


Figure 7. *Team TelunXupt:* The framework of Multi-level Dispersion Residual Network (MDRN)



(a) EADB (b) BSRB (c) BSConv

Figure 8. *Team TelunXupt:* Enhanced attention distillation block (EADB).



(a) ESA to MDSA (b) CCA to ECCA

Figure 9. *Team TelunXupt:* (a) Multi-level dispersion spatial attention (MDSA). (b) Enhanced contrast-aware channel attention (ECCA).

The shallow feature extraction is achieved by a 3×3 convolution. We then stack 8 feature extraction blocks (FEBs) to gradually refine the extracted features. Finally, the SR image is produced by the reconstruction module, which only consists of a 3×3 convolution and a sub-pixel [77] operation. The FEB block is implemented by a multi-scale fea-

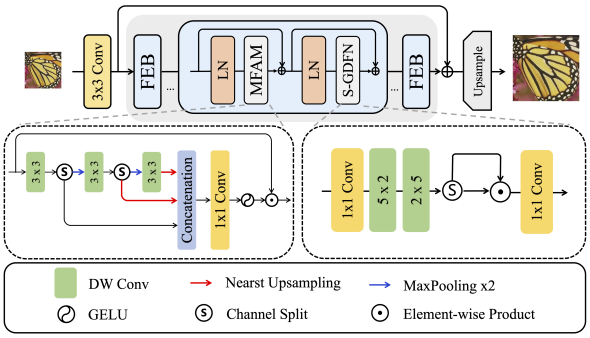


Figure 10. *Team NJUST_E:* An overview of the proposed MFA model.

ture attention module (MFAM), a simple fated-Dconv feed-forward network (S-GDFN), and two skip-connections. The number of all intermediate features is set to 36.

Training description. We first train the proposed MFA on the DIV2K [1] and Flickr2K [65] datasets. The cropped LR image size is 96 $\times$ 96 and the mini-batch size is set to 64. The MFA is trained by minimizing L1 loss and the frequency loss [11] with Adam optimizer for total 500,000 iterations. We set the initial learning rate to 1×10^{-3} and the minimum one to 1×10^{-6}, which is updated by the Cosine Annealing scheme [72].

After that, L2 loss is used for fine-tuning. The initial learning rate is set to 5×10^{-4} for 150, 000 iterations. In each training mini-batch, we randomly crop 16 patches of size 160$\times$160 from LR images as the input.

4.6. NJUST_M

General method description. The NJUST_M introduces a gated feature modulation network (GFMN) for efficient SR, which is modified from the SAFMN [81]. To make the SAFMN more lightweight, we replace the used convolutional channel mixer (CCM) with the gated-dconv feed-forward network [102] (GDFN). Fig. 11 shows that our GFMN first uses a convolution layer to map the input image to feature space and employs 8 feature mixing modules (FMMs) for learning discriminative feature representation, where each FMM block has a spatially-adaptive feature modulation (SAFM) layer and a GDFN module. To recover the HR target image, we introduce a global residual connection to learn high-frequency details and employ a lightweight upsampling layer for fast reconstruction, which only contains a 3×3 convolution and a pixel-shuffle [77] layer.

Training description. We train the proposed GFMN on the LSDIR [59] dataset. The cropped LR image size is 96 $\times$ 96 and the mini-batch size is set to 64. The SAFMN is trained by minimizing L1 loss and the frequency loss [11] with Adam optimizer for total 600,000 iterations. We set the initial learning rate to 1×10^{-3} and the minimum one to 1×10^{-6}, which is updated by the Cosine Annealing scheme [72].

4.7. KaiBai_Group

Network Architecture: The network architecture continues the design of EFDN [90] but removes the skipped connection. The plain design decreases the model's depth and complexity, as exhibited in Fig. 12.

Reparameterizable Convolution: The reparameterization technique plays a significant role in improving the performance of lightweight CNN-based methods. In PFDN, the KaiBai_Group combine the existing RRRB and a layer-wise loss based on normalized cross-correlation to enhance the layer-wise representation.

Partial Feature Distillation: Driven by [52], the KaiBai Group designed a partial local feature distillation block, dubbed PFDB. The main idea is that the intermediate features share high similarities among different channels [34]. This allows the network to process partial features in the middle layer, which can reduce the parameters and FLOPs as well as memory access.

Implementation details: To obtain the LR-HR image pairs, the KaiBai_Group leverage bicubic interpolation to downscale the 2K resolution images from DIV2K and Flickr2K. They augment the training datasets by horizontal flips and 90° rotations. The HR path size and mini-batch size are determined by the training step. The training procedure can be summarized as follows.

1. Training from scratch. The LR patch size is set to 64×64, and the mini-batch size is 96. $\mathcal{L}_1$ loss and Adam optimizer are utilized in optimization. The learning rate is initialized as 5×10^{-4} and halved at $\{250k, 400k, 450k, 475k\}$. The total number of iterations is 500k.

2. Repeat training with larger patches. The LR patch size is sequentially set to 128×128, 160×160, 180×180, and the initial learning rate is 2×10^{-4}. We reparameterize the model before the 180×180 step.

3. Fine-tuning. The LR patch size and mini-batch size are 240×240 and 128, respectively. The $\mathcal{L}_2$ loss is chosen to promote PSNR value. The learning rate is 1×10^{-5}.

The proposed method is implemented under the PyTorch framework with 4 NVIDIA RTX 3090 GPUs.

4.8. NoahTerminalCV A

Computational complexity and memory consumption are crucial aspects for efficient super-resolution since edge

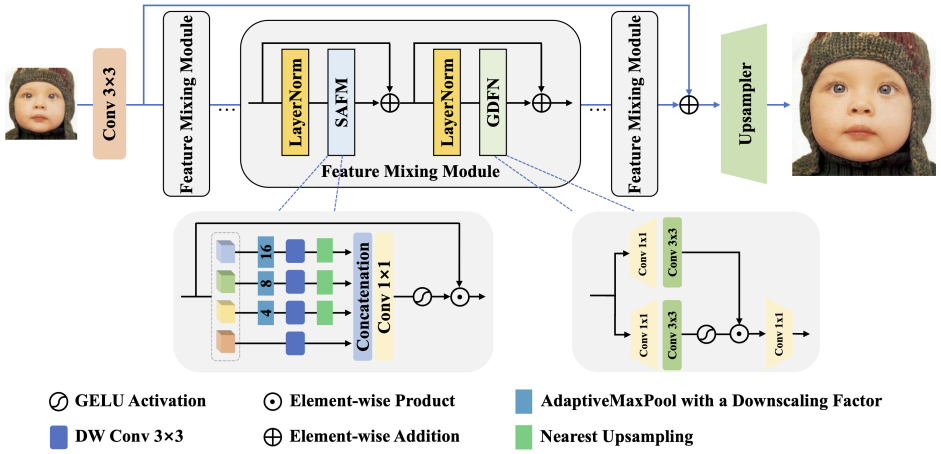


Figure 11. *Team NJUST_M:* An overview of the proposed GFMN.

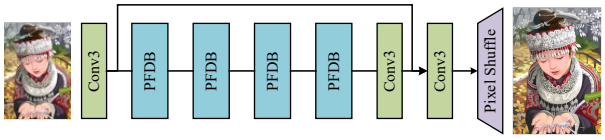


Figure 12. *Team KaiBai Group:* Network architecture of the proposed PFDN.

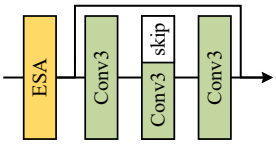


Figure 13. *Team KaiBai Group:* Partial Feature Distillation Block.

devices used for deployment are resource-constrained. In order to decrease the runtime overhead our team proposes a NoahEfficientSR (NESR) network. Inspired by the recent advancements in efficient super-resolution [42, 52, 60, 66], our model follows one of the canonical SR architectures (see Fig. 14) and consists of 4 residual edge-oriented convolutional blocks (RECB). The RECB includes 3 convolutions followed by simplified enhanced spatial attention (SESA) block.

Attention Mechanism. We further optimized enhanced spatial attention block [52] and propose a simplified version, called SESA (see Fig. 15). Moreover, we empirically found that removing the attention part in some blocks does not lead to the performance drop, whereas it decreases runtime and the number of parameters. Thus, for each second RECB, we use a single SESA block: orange RECB has no attention in Fig. 14.

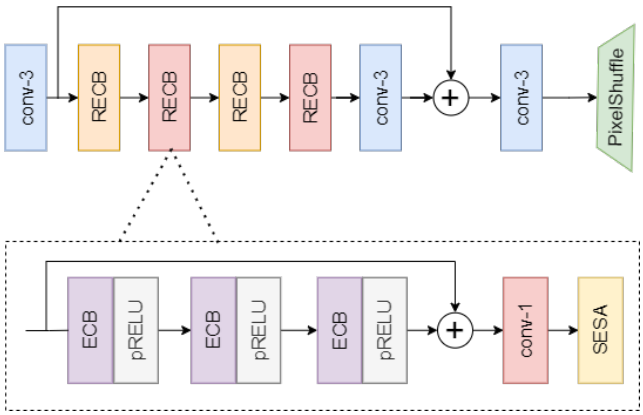


Figure 14. *Team NoahTerminalCV A:* The overall architecture of the NoahEfficientSR network.

Reparameterization. A reparameterization technique aims to bring a performance gain, due to training stabilization and doesn't change a computational complexity during the inference. Therefore, we have explored several reparameterization blocks from traditional skip-connection [4], ERB [26], ECB [106], to a self-implemented combination of ERB [26] and ECB [106]. We achieved the best PSNR performance with Edge-oriented Convolutional Block (ECB) [106] that includes 5 parallel Conv2D with different initialization during the training phase and merges into a single Conv2D layer during inference.

Implementation details. The proposed NESR has 4 RECB

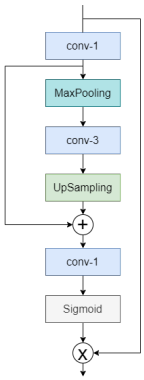


Figure 15. *Team NoahTerminalCV A:* Proposed Simplified Enhanced Spatial Attention (SESA) Block.

blocks and each has 48 feature channels, while channels in SESA are reduced to 16. We employ DIV2K [83] and LSDIR [60] datasets and 8 Nvidia V100 to train our model. The training strategy consists of several steps:

- Firstly, NESR is trained for 10^6 iterations by minimizing L_1 loss with Adam [47] optimizer. The input patch size is set to 64×64, while a batch is 64. Input patches are randomly cropped and augmented (flip, rotate) in this and all other training stages. We utilize a Cosine Annealing scheduler with a warm-up strategy, where the learning rate linearly increases from 0 to 5e-4 during 5000 iterations and then halves every 200,000 iterations.

- In the second stage, we increase the batch size to 256 and patch size to 128. Then, the model is trained for 10^6 iterations with the initial learning rate of 2e-4 cosine decay rate of 0.8 (every 50,000 iterations and 10^4 iterations for warm-up).

- At the third stage inspired by [97] we employ a large batch training strategy: batch size increased to 968 while LR patch size remains 128×128. Since the batch size is large, we set the learning rate to 8e-4 with cosine decay. Moreover, we minimize $L_1 + L_2$ objective to optimize model parameters. At this stage, the model was trained for 200,000 iterations.

Finally, each ECB block in NESR is merged into a single convolution layer. Moreover, to further decrease the runtime by 1.5ms without performance degradation, we utilize adaptive precision control.

4.9. NoahTerminalCV B

Inspired by recent advancements of light super-resolution models [42, 52, 60, 66], we propose a mobile NoahEfficientSR (MobileNoahESR) network that includes 4 residual edge-oriented convolutional blocks (RECB) and global skip connection (see Fig. 14). RECB contains 3 Edge-oriented Convolutional Blocks [106], which is a reparameterization block and eventually will be converted into a single convolution during inference and ESA [52] based attention block (see Fig. 15). It is important to note that every other RECB includes an attention module in the MobileNoahESR. We observe no performance decrease in that case.

Kernel Decomposition. Although we designed MobileNoahESR such that it has less runtime than existing state-of-the-art RLFN [52], it still contains several times more parameters than existing light transformers [60]. Thus, the primary contribution of our method is to decompose each convolutional layer in the MobileNoahESR and thereby reduce the trainable parameters. As a kernel decomposition algorithm, we employ the Tucker Decomposition [85], which decreased parameters by 25-30%.

Implementation details. Proposed MobileNoahESR and NESR (proposed by Team A) models share the same number of blocks and features during the first 2 stages. Both networks employ DIV2K [1] and LSDIR [60] datasets and 8 Nvidia V100 for training. The training strategy contains several steps:

- The third stage starts with merging each ECB block into a single convolution and performing kernel decomposition of those layers. The model is trained for 100,000 iterations with cosine decay (init. learning rate 2e-4) and batch size of 64.

- At the final stage, we fine-tune our model with a large batch of 1024 and use LAMB [97] optimizer. Moreover, we set the learning rate to 4e-4 with cosine decay and minimize $L_1 + L_2$ objective. At this stage, the model was trained for 300,000 iterations.

4.10. SeaOuter

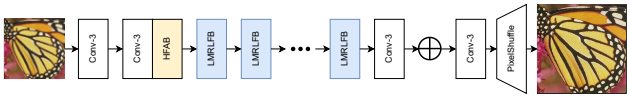


Figure 16. *Team SeaOuter:* The architecture of low memory residual local feature distillation network (LMRLFN).

Network Architecture. The SeaOuter team proposes an efficient method for super-resolution called Low Memory Residual Local Feature Network (LMRLFN). LMRLFN employs a basic SR architecture similar to RLFN [51] and FMEN [26], as illustrated in Fig. 16. However, LMRLFN uses LMRLFB blocks with a smaller number of parameters and floating-point operations (FLOPs) than the previous works, combining the advantages of both models.

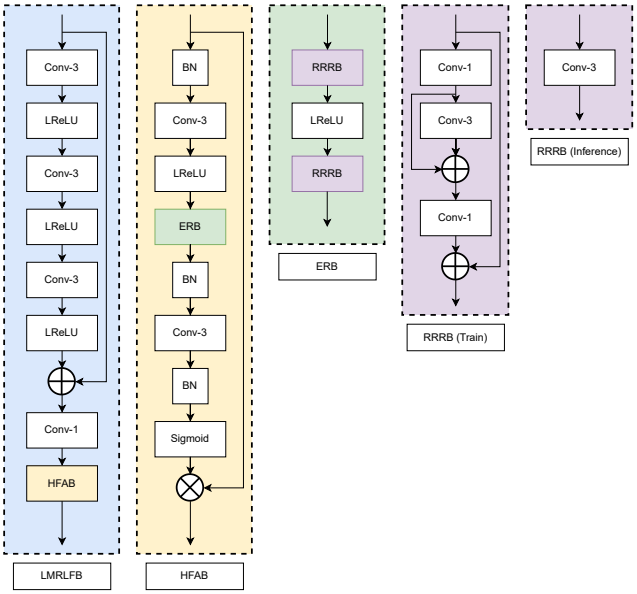


Figure 17. *Team SeaOuter:* The architecture of low memory residual local feature distillation block (LMRLFB).

LMRLFB is a modified version of residual local feature block (RLFB), as shown in Fig. 17. The LMRLFB in the proposed model not only incorporates the HFAB [26] technique within the LMRLFB structure but also employs residual learning to maintain finer details and further enhance the network's overall performance. Moreover, LMRLFB reduces the number of parameters to reduce inference time and memory while improving the performance of the model using various techniques such as wavelet loss and a warm-start strategy during fine-tuning, with different patch sizes.

Wavelet Loss. The wavelet transform loss is a mathematical tool that has shown impressive performance in self-supervised learning [41] and even knowledge distillation in image classification [105]. The discrete wavelet transform (DWT) is a pyramidal image decomposition technique that is useful for capturing both spatial and frequency information in an image. With DWT, each image can be decomposed into four bands: LL, LH, HL, and HH, where LL indicates the low-frequency band and the others are high-frequency bands. It helps to minimize the difference in high-frequency information between the ground truth and the output from the model. The loss function used for this transformation, denoted as L_W, is formulated as follows:

$$L_W = \frac{1}{n}\sum_{i}^{n} ||(\Psi^H \circ f_{gt})(x_i) - (\Psi^H \circ f_{sr})(x_i)||_1. \quad (5)$$

Here, N is the number of training samples, and Ψ^H represents the DWT operator for high frequency. After decomposing high frequency through the DWT operator, ℓ_1 norm is used to measure the difference between the high frequency of the ground truth and the output from the model.

Warm Start Strategy. The previous studies [51, 67] utilized warm-start training approaches to enhance the performance of their models or to train for larger scale factors, but they excluded the fine-tuning stage in their implementation of the warm-start strategy. We find that implementing the warm-start strategy in the fine-tuning stage can further improve the performance of the model. The training settings, such as the patch size, batch size, and learning rate, remain unchanged in the first three stages of the model training. However, the fourth and final stage uses a patch size of 640×640, a batch size of 64, and a learning rate of 1×10^{-5} with L1 loss. In the final training stage with warm-start strategy, the patch size is increased to 1024×1024, the batch size is reduced to 32 due to GPU memory limitations, and the learning rate remains the same as in the fourth stage.

Implementation details of LMRLFN. The model is trained using two datasets: DIV2K [1] and LSDIR [59]. Data augmentation techniques such as random flipping and 90-degree rotations are applied to increase the amount of data available for training. Low-resolution (LR) images are created by reducing the size of high-resolution (HR) images using bicubic interpolation. The training process consists of five stages. During the training process, HR patches of size 256×256 are randomly cropped as the ground truth, and a batch size of 64 is used. In the first stage, the model is trained from scratch. Then, the warm-start strategy is employed twice, using the model weights from the previous stage as the starting point for the current stage. The L1 loss and Wavelet transform loss are minimized using the Adam optimizer with specific hyperparameters. The learning rate is set to 5×10^{-4} and is decreased by half every 1×10^5 iterations. In the fourth training stage, the model is fine-tuned using the L2 loss with HR patches of size 640x640 for 500K iterations. Finally, the model is further fine-tuned with the warm-start strategy using larger HR patches of size 1024×1024 for another 500K iterations. The learning rate during the fine-tuning stage is set to 1×10^{-5}.

4.11. Antins_cv

ERLFN. Our method is built on Residual Local Feature Network (RLFN) [52]. Based on this network, we prune the architecture and introduce the Enhanced Residual Block (ERB) RepBlock proposed by [60] the runner-up solution, and we propose our Enhanced Residual Local Feature Network (ERLFN).

Shrinked RLFN. The RLFN proposed by [52] is an efficient network for lightweight super-resolution task. While for this efficient super-resolution task, we further prune the

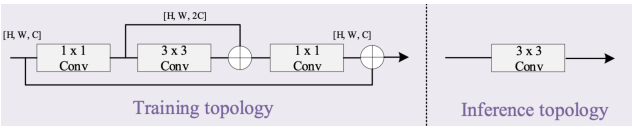


Figure 18. ERB RepBlock

network for an ideal speed. We retain the four RLFB blocks in RLFN while shrinking the model by removing ESA module nested in the first and third RLFB. The channel of the remained ESA modules is 16.

ERB RepBlock. We apply the RepBlock in the Enhanced Residual Block (ERB) first proposed by [60] the runner-up solution. We replace the 3×3 convolutions in RLFB with the ERB RepBlock.

As shown in Fig. 18, during training, the ERB RepBlock first expands the feature channels by a 1×1 convolution. A 3×3 convolution with a short skip connection then extracts features in a higher dimensional space. And a following 1×1 convolution reduces features to the original channel numbers. A skip connect eases the feature learning procedure. During inference, this RepBlock can be merged to a single 3×3 convolution.

Training Strategy We train our ERLFN model in two stages.

In the first stage, we train our model from scratch on DIV2K and full LSDIR datasets. The HR images are randomly cropped to patches of size 256×256. We adopt Adam optimizer with L1 loss for this stage. We set the initial learning rate to 5×10^{-4}, with a mini-batch size of 64, train the model for 1000 epochs, and decay the learning rate by 0.5 every 200 epochs. It takes about 34 hours for training State 1, running on four NVIDIA V100 GPUs.

In the second stage, the model is initialized with the pre-trained weights from the first stage. We finetune the model using a cosine learning rate schedule with an initial learning rate of 1×10^{-4} for 500 epochs, on DIV2K and a subset of LSDIR. The L2 loss is applied. It takes about 14 hours for training at this stage.

For inference, the ERB RepBlock is reparameterized to a 3×3 convolution. We acquire no performance drop after reparameterization. We obtain an online validation PSNR of 29.00 and a test PSNR of 26.95.

4.12. Young

General method description. The overall architecture of their network is shown in Fig. 19, which is inspired by previous leading methods [26, 50]. They propose a channel-aware re-parameterization network, composed of three parts: feature expanding, feature learning, and reconstruction part. Due to its compact and plain features, it can reconstruct high-fidelity images with fast speed. Meanwhile, in the last part, they add an auxiliary head with an $\times 2$ scaling factor that served as an additional supervised signal. In the feature learning part, they carefully optimize the components of RLFB and proposed CARB. The residual connection is widely adopted in network design. Apart from its good performance, it slows down the inference speed of the network owing to its high MAC (memory access cost). So they remove the skip connection and apply the re-parameterization technique. Specifically, they replace 3×3 convolution by RRRB [26] during the training phase as shown in Fig. 20. This is similar to the treatment in FMEN [26]. Besides, Yolov7 [88] also reports that jointly applying skip connection and re-parameterization techniques leads to the degradation of performance. Second, they simply remove 1×1 convolution in RLFB. Then they replace the ESA module with SE module [39] which is speed-friendly.

Training description. They adopt DIV2K [1], Flickr2K and LSDIR [59] as training datasets. And they train the network on RGB channels and augment the training data with random flipping, rotations, and channel shuffling [103]. The channel of each 3×3 convolution is set to 64 and 64. The training process contains 2 stages. In the first stage, they randomly crop HR patches of 256×256 from ground truth and the batch size is set to 32. They adopt AdamW optimizer by setting $\beta_1 = 0.9$, $\beta_2 = 0.9$ and $\lambda = 1 \times 10^{-3}$ and minimize L1 loss.The auxiliary head is used to help train. The initial learning rate is set to 1×10^{-3} and equipped with cosine learning rate decay. The iterations are set to $1 \times^{6}$ in the first stage. In the second stage, they remove the auxiliary head and minimize L2 loss for $1 \times^{6}$ iterations. The initial learning rate is set to 1×10^{-4}. And they increase the HR patch size to 512×512.

4.13. NTU607_ESR

Our model is based on RLFN [50] and the number of channels is compressed in order to reduce FLOPS and the number of model parameters.

The training process is divided into three stages in total. Throughout the entire training process, we use the Adam optimizer with $\beta = (0.9, 0.999)$, $\epsilon = 1e-8$, and train for 1200 epochs in each stage. Weight decay is not applied. For the training images, we perform the random horizontal flips and the random crops, except for the last stage where we do not perform random crop. In the first stage, we use the L1 loss for the model optimization with a batch size of 32, Also, we adopt random cropping images and set the patch size to 256×256. In the next stage, we use the PSNR loss with a patch size 512×512 and the warm-up training. Finally, in the third stage, we continue to use the PSNR loss, but the batch size is changed to 1, and no random cropping is applied to images. Instead, we use the complete images

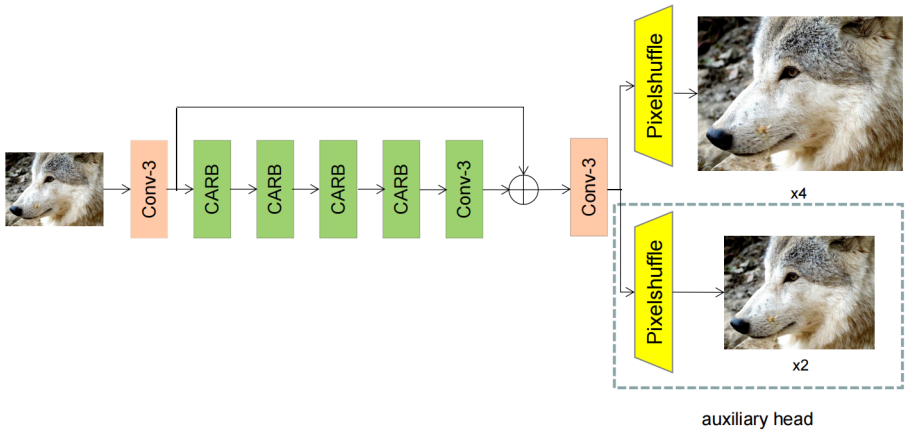


Figure 19. *Team Young:* The overall architecture of CARN.

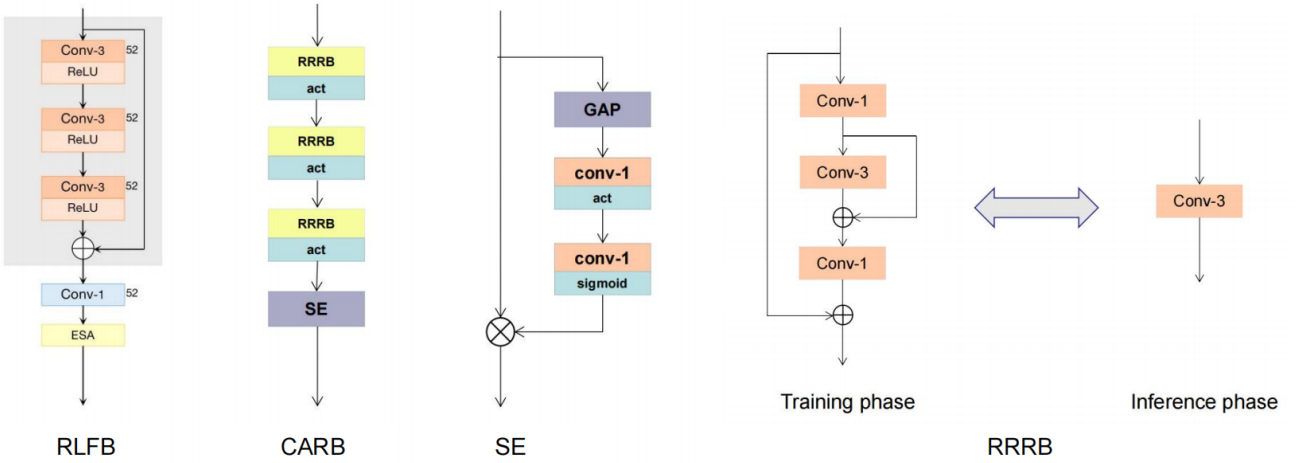


Figure 20. *Team Young:* An overview of the proposed CARB model

.

for training and also include the warm-up training. Regarding the learning rate, in the first stage, the initial learning rate is set to 5×10^{-4}, and the learning rate is decayed by half at epochs [200, 400, 600, 800, 1000]. For the second and third stages, we set the initial learning rate to 5×10^{-7} and warm it up for 100 epochs until it reaches 4×10^{-5}. Then, we apply the same decay strategy which is the same as that in the first stage.

4.14. CMVG

General method description. We propose a knowledge distillation and re-parameterization(KDRP) efficient SR network, which is an advanced version of RLFN [51] model that gets the best inference time in NTIRE 2022 Efficient Super-Resolution Challenge. We introduce knowledge distillation, and re-parameterization, as well as replace the ReLU activation function with GeLU activation and reduce network width to further compress the model, but also maintain the performance. The framework of our KDRP is shown in Fig. 21, which is designed based on the knowledge distillation SR network [53]. KDRP model is composed of the teacher SR network and the student SR network, the student network is an end-to-end re-parameterization residual(RPSR) network that recovers SR images. The teacher network super-reconstructs HR images in an Encoder-Decoder pattern. Due to the teacher network using HR images as privileged information to reconstruct HR images, it learns more high-frequency information. We achieve knowledge transfer through knowledge distillation, which provides extra rich detailed information for the student network.

The encoder of the teacher network extracts features of input HR images and down-samples them with a factor of 4 to get reconstructed LR images. The encoder learns the degradation process of the HR image, and its structure is the same as the encoder in [53]. The decoder structure is the same as the student network, which is used to recover HR images from reconstructed LR images. The student network includes the shallow feature extraction, the depth feature extraction, and the SR reconstruction module. There are 4 re-parameterization residual blocks(RPRB) in the depth feature extraction. The framework of RPRB

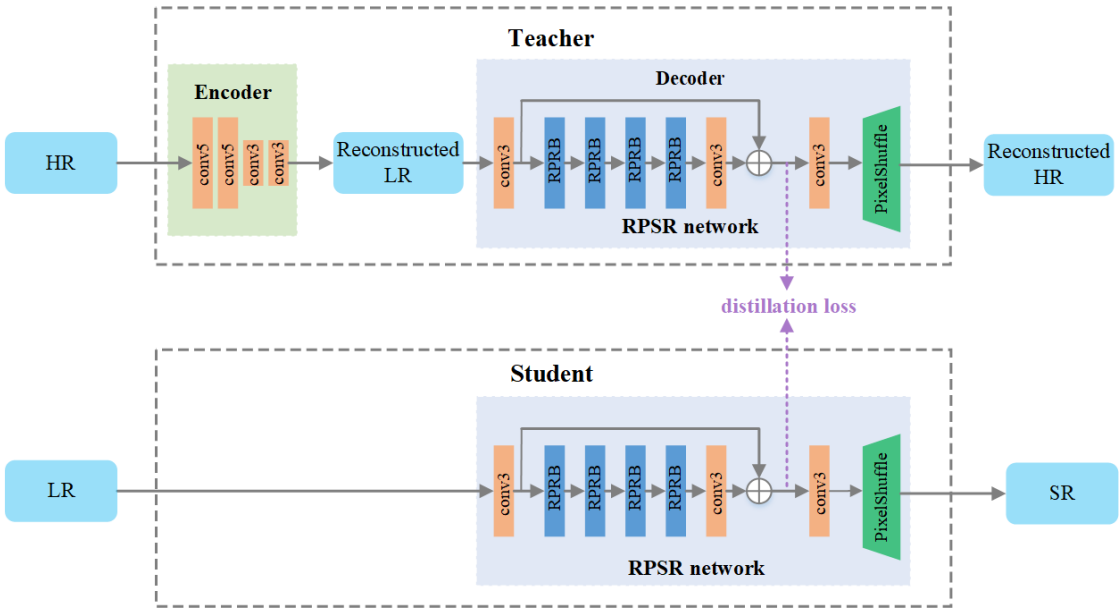


Figure 21. *Team CMVG:* The framework of KDRP

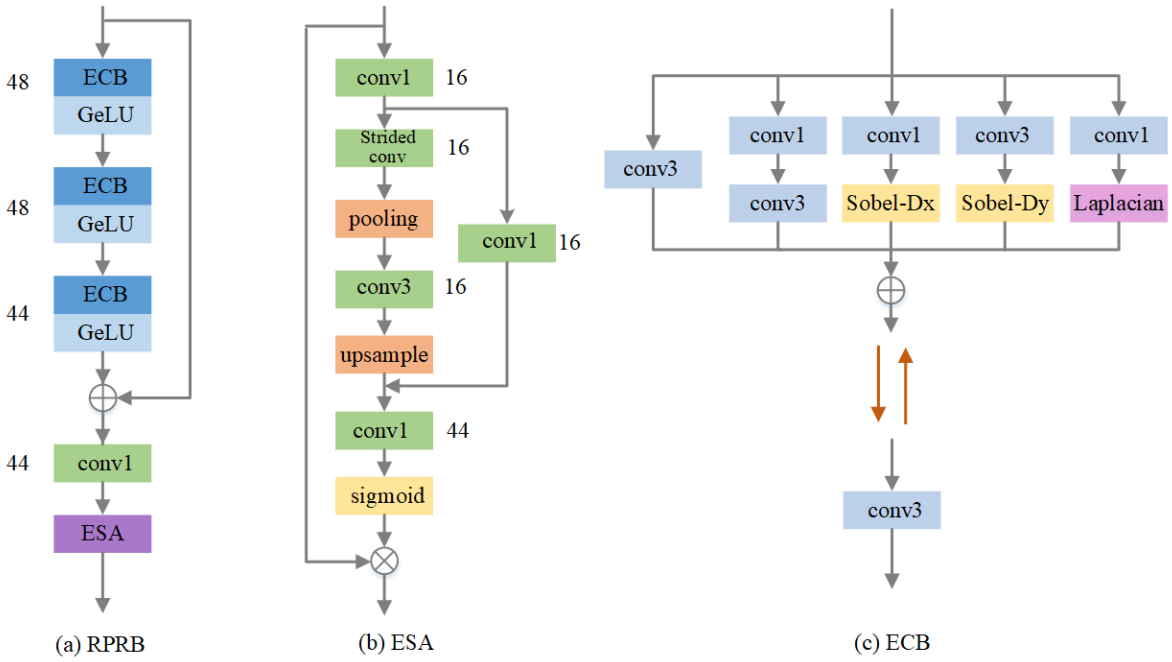


Figure 22. *Team CMVG:* The framework of RPRB

is shown in Fig. 22, which is composed of 3 residual connected ECB [107]+GeLU blocks and ESA [51] . We utilize ECB blocks during the training phase, while it can be equivalent to a convolution layer in the inference phase. The use of re-parameterization can increase the model expression ability and boost the SR performance, but not cost more inference time. The number of feature maps of Conv3 in RPSR model is set to 44, 44, and 48 from left to right, and the number of feature maps in RPRB is marked in Fig. 22.

Loss function. In the teacher network, since the encoder simulates the mapping relationship between HR-LR image

pairs, we use the imitation loss that calculates L1 distance to constrain encoder training. In order to ensure that the teacher network can accurately recover HR images, it is also necessary to constrain the teacher network using reconstruction loss. The total loss for optimizing the teacher network is defined as follows:

$$\begin{aligned} l_{teacher} &= l_{recon}^{HR} + l_{imitation} \\ &= \left\| Y_T^{SR} - Y \right\| + \left\| X_E - X \right\| \end{aligned} \tag{6}$$

where X is the ground truth LR image, Y is the HR image, the output of encoder and decoder is X_E and Y_T^{SR} respectively, $\|.\|$ denotes the L1 distance.

We use the distillation loss [53] and reconstruction loss to train the student network, the distillation loss is calculated through the features of the penultimate convolution layer of the teacher and student network, and the total loss is defined as:

$$\begin{aligned} l_{student} &= l_{recon}^{SR} + 0.001 * l_{distillation} \\ &= \left\| Y_S^{SR} - Y \right\| + 0.001 * l_{distillation} \end{aligned} \tag{7}$$

Training details. We use the DIV2K [1], Flickr2K, and LSDIR [59] datasets to train the proposed KDRP model, the training details are described as follows:

Stage 1. Training teacher network. The teacher network is trained from scratch. The 256×256 HR patches are randomly cropped from HR images, and the batch size is set to 64. The teacher model is trained by teacher loss with Adam optimizer. The initial learning rate is set to 5×10^{-4} and halved at every 60 epochs. The total number of epochs is 400.

Stage 2. Training student network. We use the pretrained weight from the teacher network to initialize the student network, and then train the student network with teacher loss. Other parameter settings are kept the same with Stage 1.

Stage 3. Fine-tuning student network. The fine-tuning steps are described as follows:

- The student model is initialized from Stage 2 and trained with the same settings as Stage 1. In particular, the loss function is only the L1 loss.
- The student model is fine-tuned by MSE loss further, and trained with the same setting as Stage 3.1.
- HR patch size is set to 512×512 and the student model is trained with MSE loss.
- The student model is fine-tuned on DIV2K and Flickr2K dataset with 640×640 HR patches and MSE loss. The initial learning rate is set to 2.5×10^{-4} and halved at every 100 epochs. The total number of epochs is 500.

After finishing the training of the KDRP model, we only use the student network to reconstruct SR images during the inference phase, at the same time, the multiple branches of the ECB module can be merged into a convolution layer.

4.15. Touch_Fish

General method description. The Touch_Fish team proposes an efficient method for super-resolution called AttLi. The rationale behind utilizing an attention map with a considerable perception field is that it can be advantageous for the preceding layers to concentrate their attention on regions of interest. Thus, they generate an attention map $\mathcal{M}(i,j)$ as follows,

$$\mathcal{M}(i,j) = \phi(\text{Conv}_{1\times1}(F_l(i,j))), \tag{8}$$

where $\phi(\cdot)$ denotes the sigmoid function. $F_l(i,j)$ and $F_f(i,j)$ denote the value of the feature map in the position (i,j) from the latter layer and former layers, respectively. Then they use the generated attention map to reweight the features in the former layers as,

$$\mathcal{M}(i,j) \bigodot F_f(i,j), \tag{9}$$

where $\bigodot$ denotes the Hadamard product.

As depicted in Fig. 23(b), an attention map is generated for each block, which is subsequently utilized to reweight the feature maps originating from distinct levels. The present method features the integration of attention maps at three distinct levels, with the aim of directing the feature maps of lower layers toward the effective modeling of remote dependencies.

Furthermore, they have implemented the reparameterization technique (rep) [22] to enhance the efficiency of the inference phase. The rep methodology has been incorporated into each convolutional block depicted in Fig. 23(a). This approach involves utilizing the Hadamard product to encode the short-range correlation between the feature maps in the input of the block and the feature maps generated after four convolutional layers in the output of the block.

Efficiency Improvement. In contrast to prior techniques, they have replaced the employment of stride convolutions, pooling, and upsampling with merely the use of a generated mask. This modification has resulted in a significant acceleration of both inference and training times, as well as a reduction in the memory footprint. The effectiveness and efficiency of this mask arise from its capacity to facilitate attention from the subsequent layers to guide the learning process of the earlier layers.

Implementation details. The dataset utilized for training comprises of DIV2K and LSDIR. The training procedure

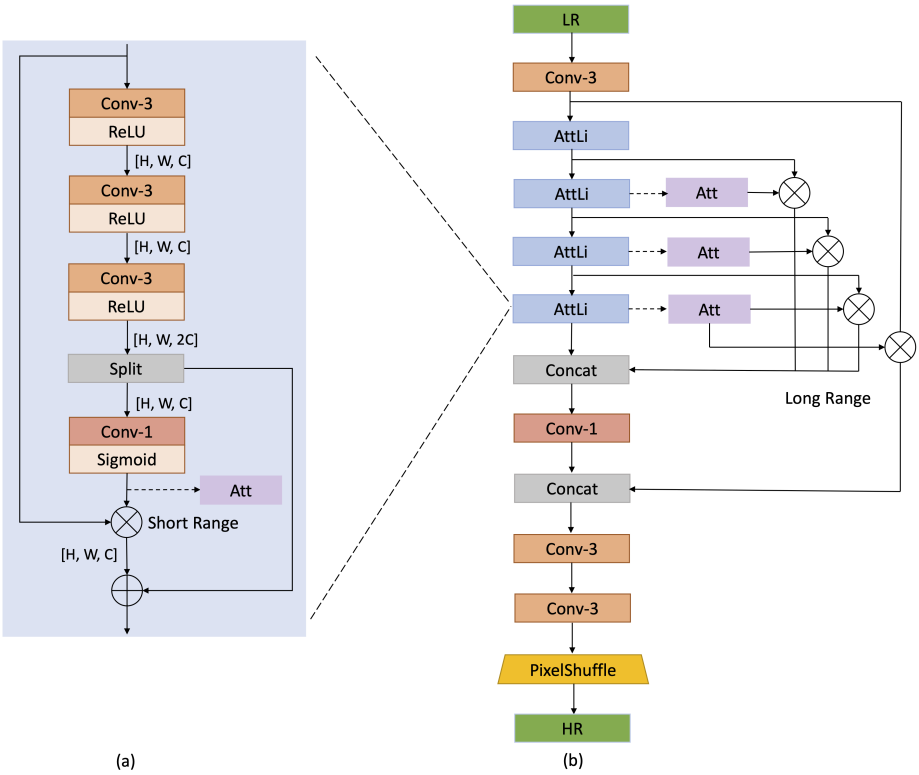


Figure 23. *Team Touch_Fish:* (a) AttLi block. Att in pink denotes the generated attention map $\mathcal{M}$. (b) Pipeline. Input is the low-resolution image and output is the high-resolution image.

entails the deployment of four AttLi modules arranged in a sequence, each consisting of 48 feature maps. During each training batch, 64 HR RGB patches are cropped, measuring 256×256, and subjected to random flipping and rotation. In the training phase, NGswin [13] is used as the teacher model for boosting the restoration performance. The learning rate is initialized at 5×10^{-4} and undergoes a halving process every 2×10^5 iterations. The network undergoes training for a total of 10^6 iterations, with the L1 loss function being minimized through the utilization of the Adam optimizer [47]. they repeated the aforementioned training settings four times after loading the trained weights. Subsequently, fine-tuning is executed using the L1 and L2 loss functions, with an initial learning rate of 1×10^{-5} for 5×10^5 iterations, and HR patch size of 512. They conducted fine-tuning on four models utilizing both L1 and L2 losses, and employed batch sizes of 64 and 128. Finally, they integrated these four models to obtain the ultimate model.

4.16. CUC_SR

General method description. The CUC_SR team proposed a Reparameterized Residual Feature Network (RepRFN) [18] as shown in the Fig. 24. This work was inspired by RFDN [66], RLFN [52], and ECBSR [107], the team found that the overall structure of RLFN and ECBSR is similar, they both remove the operation of channel con-

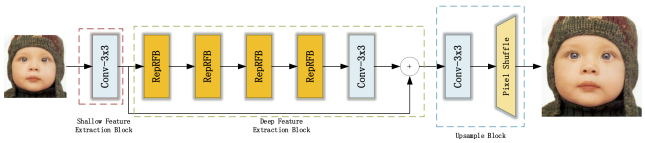


Figure 24. *Team CUC_SR:* The structure of RepRFN.

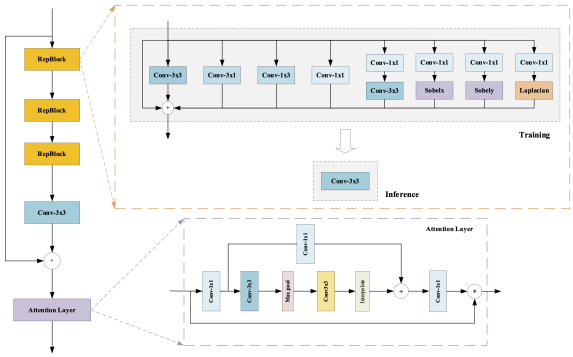


Figure 25. *Team CUC_SR:* The structure of RepBlock.

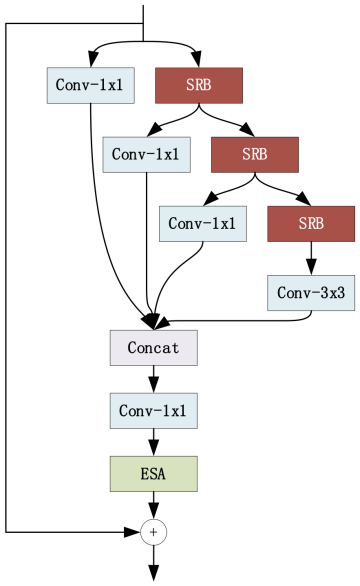


Figure 26. *Team CUC_SR:* The structure of RFDB.

catenation. In fact, in a simple experiment, if the channel concatenation in RFDN were replaced by local residual connections and then retrained with DIV2K [1] training set, although the number of parameters and FLOPs increased slightly, the overall PSNR was relatively similar, and the inference speed could be improved by about 34% and maximum GPU memory consumed during inference could be decreased by about 55%. It can be considered that channel concatenation is a factor affecting the speed of inference. Therefore, the team rethought RFDN and mainly improved the following parts.

First, parallel SRB and 1×1 convolution in RFDB are replaced by 3×3 convolution, and channel concatenation is replaced by local residual connection. However, in consideration of the simple receptive field of 3×3 convolution, inspired by the recent reparameterization works [19, 20, 107], in order to capture features of more patterns as much as possible, as shown in the Fig. 25, the team design a reparameterized multi-branch Block called RepBlock to extract features and use residual connections for features fusion. In the training stage, the multi-branch structure is used, and the model is converted into a simple plane structure through structural reparameterization to speed up inference.

In RFDN, as shown in the Fig. 26, features are distilled by SRB and 1×1 convolution three times in each RFDB, so the team replaces the first three 3×3 convolution with this multi-branch structure. Considering that some 1×1 convolution in RFDN is to perform channel transformation after channel concatenation, it doesn't need to perform channel transformation due to residual connection, so these 1×1 convolutions can be removed to further compress parameters. Inspired by RLFN [52], the convolution group in ESA [66] is also reduced to one layer of convolution. The final building block structure of RepRFB is shown in the Fig. 25.

In addition, a loss function based on Fourier transform is also used. Specifically, the loss of the SR image and HR image after Fourier transform is calculated to guide the model to learn frequency information. The loss can be formulated as:

$$L(x, y) = L1(fft(x), fft(y)) \tag{10}$$

where $L1$ defines L1 loss, $fft(\cdot)$ means Fast Fourier Transform on the image. It should be noted that we only perform Fourier transform on the scale dimension of the image.

Training strategy. In terms of training, DIV2K training set and Flickr2K dataset are used, and the HR patch size is set to 192×192. Random horizontal flip, vertical flip, and rotation are introduced into the data augmentation during training. The proposed RepRFN consists of 4 RepRFBs, and the number of channels is set to 48. The model is trained from scratch. Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ is used. The batch size is set to 64, and the initial learning rate is set to 5×10^{-4} and halved at every 100 epochs. The total number of epochs is 1001. In the process of training, the loss function is the combination of pixel loss and the loss based on Fourier transform. In practical application, Charbonnier loss function can avoid the problem that the results generated by L1 loss function and L2 loss function are too smooth [112]. In the experiment, the team also found that the Charbonnier loss is better than the L1 loss in terms of PSNR. So they chose Charbonnier loss as pixel loss L_{pix}. Finally, the loss can be formulated as:

$$L(x, y) = \lambda_1 L_{pix}(x, y) + \lambda_2 L1(fft(x), fft(y)) \tag{11}$$

where $\lambda_1 = 0.9$ and $\lambda_2 = 0.1$. The hyperparameter ϵ^2 in Charbonnier loss is set to $10^{-}6$.

4.17. NJUST_R

General method description. The NJUST_R presents a simple yet effective model, SAFMN [81], to solve efficient SR. As shown in Fig. 27, the SAFMN consists of the following parts: a stacking of feature mixing modules (FMMs) and an upsampler layer. Specifically, we first apply a 3×3 convolution layer to transform the input LR image to feature space and generate the shallow feature F_0. Then, the multiple stacked FMMs are used to generate finer deep features from F_0 for HR image reconstruction, where an FMM layer has a spatially-adaptive feature modulation (SAFM) sub-layer and a convolutional channel mixer (CCM). To recover the HR target image, we introduce a global residual connection to learn high-frequency

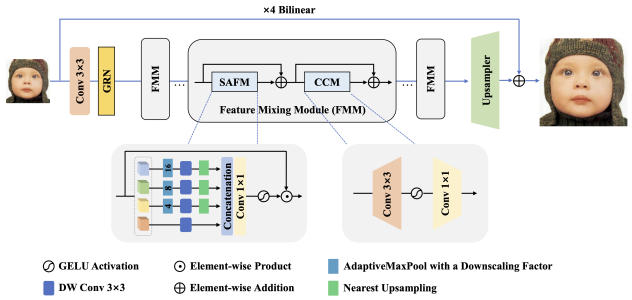


Figure 27. *Team NJUST_R:* An overview of the proposed SAFMN.

details and employ a lightweight upsampling layer for fast reconstruction, which only contains a 3×3 convolution and a pixel-shuffle layer [77].

To speed up the inference time of the original SAFMN, we made the following modifications:

- We remove all LayerNorm layers from FMMs and only normalize the features after the first convolution with the Global Response Normalization [75].
- We disable the learnable bias of all employed convolution operations.
- We equip the first and last convolution with re-parametrization technique [23].
- We change the skip-connection in feature space into image space.

Training description. We train the proposed SAFMN on the LSDIR [59] dataset. The cropped LR image size is 96×96 and the mini-batch size is set to 64. The SAFMN is trained by minimizing L1 loss and the frequency loss [11] with Adam optimizer for total 600, 000 iterations. We set the initial learning rate to 1×10^{-3} and the minimum one to 1×10^{-6}, which is updated by the Cosine Annealing scheme [72].

4.18. Sissie_Lab

General method description. The Sissie_Lab proposes Diversified Local Feature Arch-Network (DLFAN) for the efficient super-resolution task. The architecture of the model is shown in Fig. 28. The proposed DLFAN is modified from the previous work RLFN [51]. The main improvement of the approach is in two aspects. First, DLFAN uses an archway-shaped connection to merge the shallow features in recovering stage. Secondly, DLFAN enhances all the convolution layers in RLFB by using the structure re-parameterization method to extract diversified features.

Building Blocks. A variant of Diversified Local Feature Block(DLFB) is designed based on the RLFB. Inspired by re-parameterization works ECB [106] and DBB [21], a re-parameterization Learnable Enhanced Diverse Branch Block(LEDBB) is used to replace the original 3×3 convolutional layers in RLFB. Specifically, the LEDBB inside DLFB introduces diverse branches of convolution to extract different textures and edges during training. As shown in Fig. 29, there are two additional branches along with the original 3×3 convolution, which consists of a 1×1 convolution and two cascaded convolutions with sizes of 1×1 and 3×3, respectively. Besides, batch normalization is added after each convolutional layer to bring training-time non-linearity.

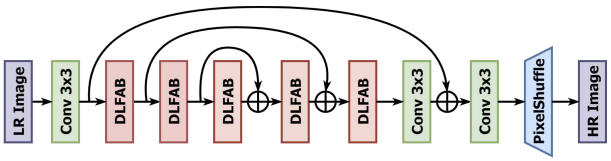


Figure 28. *Team Sissie_Lab:* The overall network framework of Diversified Local Feature Arch-Network(DLFAN).

Training strategy. The training strategies contain two stages: the training stage and fine-tuning stage.

1. **Training stage.** In each training batch, 64 HR color patches with a patch size of 256×256 are randomly cropped from HR images. The model is trained by minimizing $L1$ loss function. The initial learning rate is set to 5×10^{-4} with cosine decaying to 1×10^{-5}. The total number of epochs is 5000.
2. **Fine-tuning stage.** The HR patch size is progressively increased from 512 to 840 to improve performance. The batch size is set as 32 and 16, respectively. $L2$ loss is used in this stage. The learning rate decays from 5×10^{-5} to 1×10^{-6}. The total number of epochs in the fine-tuning stage is 3000.

DIV2K and Flickr2K are used as the training datasets. Data augmentation methods including channel shuffle, rotation, and flipping are applied. The model is trained by Adam optimizer with default settings. The model is implemented on Pytorch 1.13.1 and trained with 2 NVIDIA GeForce RTX 3090 GPUs.

4.19. GarasSjtu

General method description. The proposed mixer-based local residual network super-resolution model (MLRN) model [29] is a modified version of the model in [52] but with low computation and runtime. The concept of the MLRN model is based on convolution mixer (ConvMix) [84], and enhanced spatial attention (ESA) [68]. The ConvMix depends on depthwise convolution and pointwise

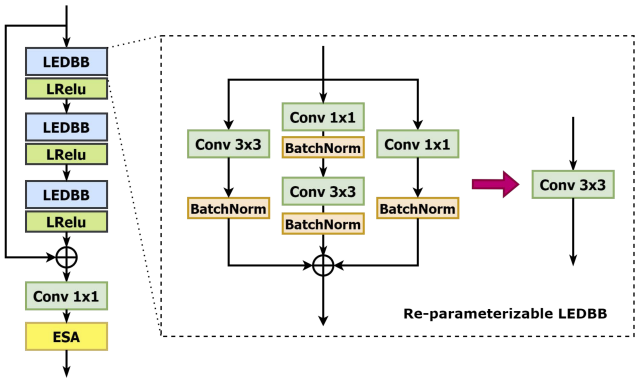


Figure 29. *Team Sissie_Lab:* Re-parameterization of Learnable Enhanced Diverse Branch Block(LEDBB).

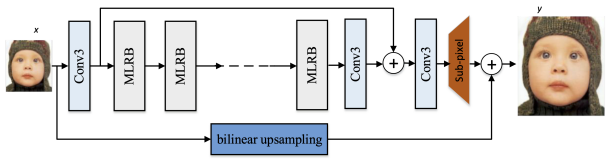


Figure 30. *Team GarasSjtu:* The structure of the proposed mixer-based local residual network (MLRN).

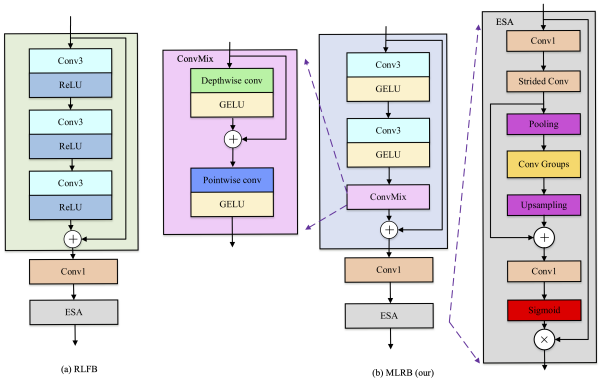


Figure 31. *Team GarasSjtu:* The structure of the proposed mixer-based local residual block (MLRB).

convolution to make spatial and channel mixing possible. While the ESA is used as a spatial attention mechanism. Additionally, we were inspired by [27, 30], using the bilinear upsampling of the low resolution (LR) at the final stage of the model.

Fig. 30 illustrates our MLRN model, which consists of three blocks, shallow feature extraction, deep feature extraction, and high-resolution image reconstruction, in addition to bilinear upsampling. Shallow feature extraction is performed by using 3×3 convolution, which changes the image domain to the feature domain.

After that, four blocks of the mixer-based local residual block (MLRB) are included in the deep feature extraction modules. We have further improved the original block in [52] by reducing weight and Multi-adds. The MLRB block is designed based on using two 3×3 convolutions and one ConvMix block with a GELU activation function, as shown in Fig. 31(b). Also, the ConvMix block is composed of a depthwise convolution and pointwise convolution using residual across the depthwise convolution as shown in Fig. 31(b). In addition, residual learning is used between the input and the output of this MLRB block. Afterward, an 1×1 convolution and ESA [68] are used at the end of the MLRB block similar to [52].

The final stage is the image reconstruction made by utilizing one Conv 3×3, also a bilinear upsampling for transferring low-frequency information. Then, our model ends with the pixel shuffle layer, which is utilized for mapping features to HR image space.

Our MLRN contains four MLRB blocks, in which we set the number of feature maps to 50. also, we set the channel number of the ESA is set to 16 similar to [68]. In our training, we used DIV2K and LSDIR [59] to train the model.

Training strategy. The training of the model is done in the following steps.

At the starting stage, we trained the model from scratch using the DIV2K and LSDIR [59] datasets, with a patch size of 256×256 and a batch size of 64. In this training, the L1 loss function is used with the Adam optimizer. This stage is trained for 800 epochs with an initial learning rate 5×10^{-4} reduced by half every 200 epochs

After the previous stage, the model starts its training from the previous pre-trained weights using the DIV2K and Flickr2K datasets with an initial learning rate 5×10^{-4}that drops by 50% at every 200 epochs for 1000 epochs using L1 loss.

4.20. USTC_ESR

Network Architecture The USTC_ESR team proposes the enhanced fast residual network (EFRNet) for efficient image super-resolution. The overall architecture is shown in Fig. 32, which is based on the previous method [50]. The difference is EFRNet uses five enhanced fast residual blocks (EFRB) as the basic building block. Moreover, we adopt the global skip connection at the image level. Specifically, EFRB also employs GELU [35] as the activation function. In addition, to achieve efficient feature extraction, we integrate the partial convolution (PConv) [7] into EFRB. PConv reveals the feature redundancy in the existing networks and only executes the convolution operations on only a few feature channels while leaving the remaining other channels untouched, thus reducing the redundant computation and

memory cost. Besides, we also employ the ESA module [50, 67] along with the residual block.

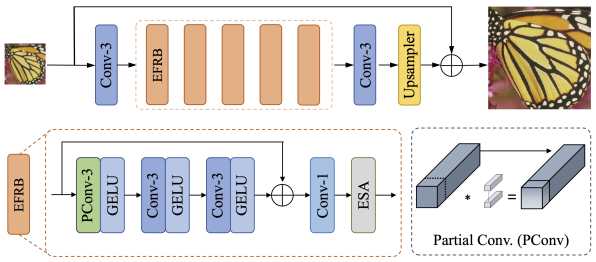


Figure 32. *USTC_ESR team*: The overall illustration of EFRNet.

Losses The framework is trained in a supervised manner with the L1 and the frequency losses between the estimated results ($\hat{\mathbf{O}}$) and the corresponding ground truth $\mathbf{Y}$, which is defined as follows:

$$\mathcal{L}_{total} = ||\hat{\mathbf{O}} - \mathbf{Y}||_1 + \lambda_{freq.} * ||FFT(\hat{\mathbf{O}}) - FFT(\mathbf{Y})||_1, \quad (12)$$

where $\lambda_{freq.}$ denotes the balanced weight and $FFT(\cdot)$ denotes the Fast Fourier Transformation.

Implementation details The experiments of EFRNet are implemented on the PyTorch platform. The model is trained by ADAM optimizer with β_1 = 0.9, β_2 = 0.999. We train the default model with the progressive training strategy. (i) At the first stage, the model is trained with the HR patch size of 192 $\times$ 192 from scratch. The initial learning rate is set to 0.0008 and halved at every 200 epochs. The total number of epochs is 600. (ii) At the second stage, the model is trained with the HR patch size of 512 $\times$ 512. The initial learning rate is set to 0.0004 and halved at every 200 epochs. The total number of epochs is 600. (iii) At the third stage, the model is trained with the HR patch size of 800 $\times$ 800. The initial learning rate is set to 0.0001 and halved at every 200 epochs. The total number of epochs is 400. For all three training stages, we adopt the same losses (Eq. (12)) as default. $\lambda_{freq.}$ is empirically set to 0.2.

4.21. SEU_CNII

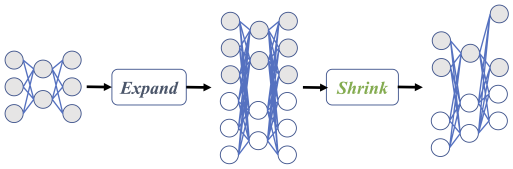


Figure 33. *Team SEU_CNII:* Designing and training a SOTA model is a non-trivial work. Our work shows that expanding an existing model and then shrinking it by pruning and finetuning is an efficient way to improve the performance of the model.

General method description. Lots of works focus on designing efficient models to achieve state-of-the-art performance on challenges year by year. A big problem about collecting more data and creating a new larger dataset for training a bigger and more powerful model is that we always need to update our model to adapt to the new dataset for learning more knowledge than before.

Efficient insights are not easy to find and some important factors could be ignored in the advanced computing platforms. Good performance in theory but bad performance in reality can happen when running MobileNet on different platforms.

We want to find a method that scales an existing SOTA efficient model efficiently and effectively. In this way, we can just absorb achievements from efficient model design research and scale them to get a new state-of-the-art in a new dataset.

We improve RFDN [66] model inspired by the "Train Big, Then Compress" theory proposed by [63] and proposed the "Expand Big, Then Shrink" idea. Big models have more parameters than small models so they can learn more knowledge from the dataset. If there is a new dataset larger than a traditional dataset which is the situation of LSDIR [59] and DIV2K [1], the old SOTA model is not enough to learn new knowledge from the new large dataset, so that we need to expand it. On the other hand, an expanded model always has redundancy in computation, using network compression method to shrink it can make it efficient and without knowledge loss.

Training details We first train RFDN on the LSDIR [59] datasets. The model is trained by Adam optimizer with 300 epochs. We set the learning rate as 1e-5. After training, heavy compression followed. We utilize a dynamic structure pruning method [28] to compress the model and finetune it iteratively for 30 epochs with a 0.8 sparsity factor. We choose the best model as the final submitted model.

4.22. AVC2_CMHI_SR

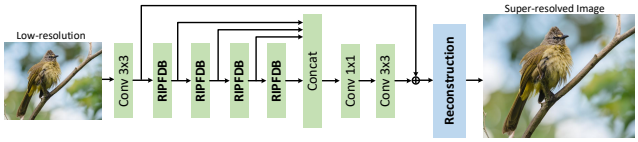


Figure 34. *Team AVC2_CMHI_SR:* The overall architecture of the proposed network.

General method description. The whole structure is shown in Fig. 34. We have revamped the Residual Feature Distillation Block (RFDB) blocks of RFDN [66] based on partial convolution [7], resulting in the key design called Residual Interactive Partial Feature Distillation

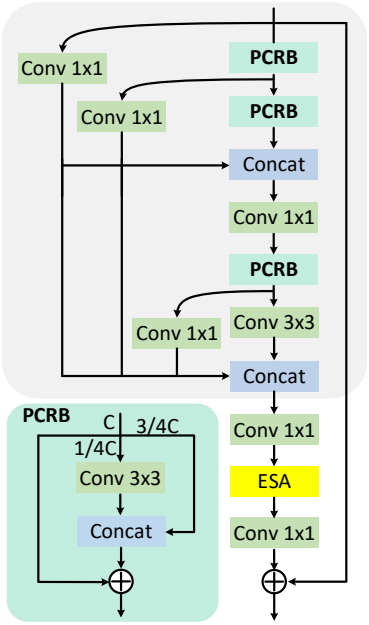


Figure 35. *Team AVC2_CMHI_SR:* The overall structure of RIPFDB.

Block (RIPFDB), which is illustrated in Fig. 35. Specifically, to further reduce complexity compared to the Shallow Residual Block (SRB) in RFDB, we introduced the Partial Convolution Residual Block (PCRB), which performs convolutional operations on only a few channels while directly concatenating the remaining channels with the feature after convolutional operation, thus significantly reducing time cost compared with depth-wise convolution. Additionally, to enhance information interaction between different distillation features, we proposed an interactive mechanism that concatenates distillation features from different stages and performs a 1×1 convolutional layer to fuse the features. To further improve performance, we applied ESA [68] to enhance spatial attention. Motivated by BSRN [62], which uses CCA [42] to improve model ability from a channel-wise perspective, we multiplied a learnable parameter along the channel direction to fully explore model capacity. In addition, we use GELU [35] as the activation layer, which has been proven to have better performance.

Training strategy. In this work, we adopt DIV2K and Flickr2K as training dataset which includes 3450 images. We employed the Adam optimizer with an initial learning rate of 5×10^{-4} to minimize the L1 loss function on patches of size 64×64 pixels. Data augmentation is performed, such as random horizontal flipping and 90° rotation. The network was first optimized for 1.2×10^6 iterations using a batch size of 64. Subsequently, the batch size was increased to 256 for an additional 5×10^5 iterations of training. The network consists of 4 RIPFDB blocks, where the partial coefficient is set to 0.25 and the number of channels is set to 52.

4.23. Set5_Baby

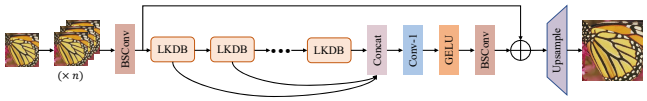


Figure 36. *Team Set5_Baby:* The overall architecture of large kernel distillation network (LKDN).

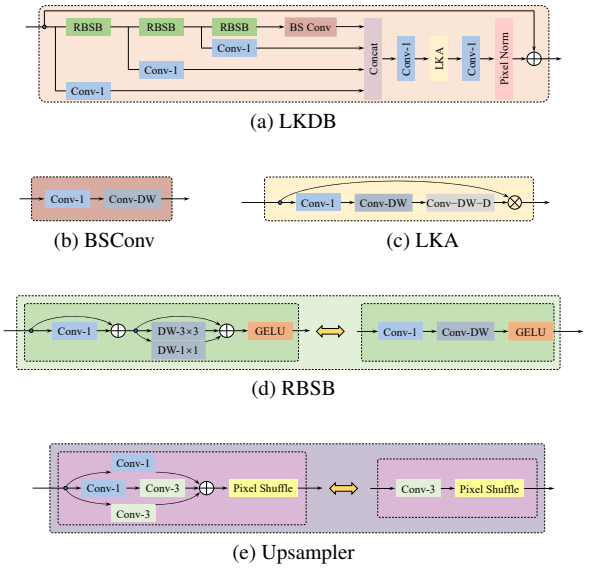


Figure 37. *Team Set5_Baby:* The details of each component. (a) LKDB: Large Kernel Distillation Block; (b) BSConv: Blueprint Separable Convolution; (c) LKA: Large Kernel Attention; (d) RBSB: Re-parameterized Blueprint Shallow Block; (e) Upsampler: Re-parameterized Upsampler.

Network Architecture. Set5_Baby team proposed Large Kernel Distillation Network (LKDN) [94], and the network architecture is as depicted in Fig. 36. It comprises four components: shallow feature extraction, multiple stacked feature distillation blocks, multi-layer feature fusion, and image reconstruction block.

The input image is replicated n times during the preprocessing stage, followed by a concatenation of the replicated images. Then the initial feature extraction is implemented by a 3×3 blueprint separable convolution [33] (BSconv) to generate shallow features from the input LR image. The structure of BSconv is shown in Fig. 37b, which consists of a 1×1 convolution and a depth-wise convolution. The next part of LKDN is to extract deep features through a stack of large kernel distillation blocks (LKDBs), which is shown in Fig. 37a. After gradually refining by the

LKDBs, all the intermediate features are fused and activated by a 1×1 convolution layer and a GELU [35] activation. A 3×3 BSConv layer is used to smooth the fused features. Finally, a skip connection is employed in the model to enhance the residual learning and the SR images are obtained through image reconstruction. The reconstruction process only includes a re-parameterized 3×3 convolution and pixel-shuffle operation [78] as shown in Fig. 37e.

The convolution decomposition is performed in a different order, and the large kernel attention (LKA) module is shown in Fig. 37c. The large 17×17 convolution is decomposing into a 1×1 point-wise convolution, a depth-wise 5×5 convolution, and a depth-wise dilation convolution with a kernel size of 5 and dilation of 3. To replace the Blueprint Shallow Residual Block (BSRB) in BSRN [62], we introduce the Re-parameterized Blueprint Shallow Block (RBSB) structure, as shown in Fig. 37d.

Implementation details of LKDN. LKDN consists of 5 LKDBs and 42 channels and is trained with RBSB. 800 images from DIV2K [1] and 2650 images from Flickr2K [65] are used as the training datasets. The training process of LKDN involves two stages: an initial training stage and a fine-tuning stage. In the initial training stage, we randomly crop 128 mini-batch HR patches with a size of 256×256. We train LKDN using the common L1 loss function with a learning rate of 5×10^{-3} and 9.5×10^{5} iterations. In the fine-tuning stage, we set the patch size of HR images and batch size to 480×480 and 64, respectively. LKDN is fine-tuned using the L_2 loss function with a learning rate of 2×10^{-5}, and a total of 5×10^{4} iterations. The exponential moving average is set to 0.999 and Adan optimizer [95], with $\beta_1 = 0.98$, $\beta_2 = 0.92$ and $\beta_3 = 0.99$ is applied in both stages.

4.24. LVGroup_HFUT

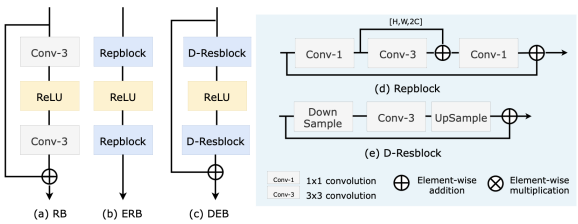


Figure 38. *Team LVGroup_HFUT:* The structure of DEB and DRB.

General method description. Considering the requirements of the Efficient Super-Resolution task, we need to improve the performance of the model as much as possible under the limited computational cost. Although FMEN can efficiently extract features while saving time and space memory, it still cannot meet the corresponding accuracy requirements. We infer that this is because it still cannot effectively extract the correlation features between pixels of low-resolution images and deep connections between different pixels of an image. At the same time, it uses relatively few residual connections to reduce computational cost consumption, which may cause losing the original feature in the process of feature extraction, resulting in reduced accuracy and unable to meet the final competition requirements. Therefore, we design an Efficient Deep Residual Network (EDRN), as shown in Fig. 39. Specifically, our backbone network is the same as FMEN [26]. At the same time, in order to better extract the deep features of the image, we modified the Repblock, designed a D-Resblock, and used a residual connection to preserve the original features of the image, which is shown in Fig. 38.

Training strategy. We trained a total of 300 epochs to bring the model to convergence using NVIDIA 3090ti with 24GB memory. The number of feature maps of DRB and DAB is set to 64 and 16, respectively. Our initial learning rate is set to 2×10^{-4}, and the learning rate decays to 4×10^{-5} after 200 epochs. Meanwhile, in the first 250 epochs, we use L1 loss, and in the last 50 epochs, we use L2 loss. Our other settings are exactly the same as FMEN.

4.25. FRL Team 4

General method description. We follow the general structure of RFDN [66], which is divided into 4 stages: shallow feature extraction, deep feature extraction, multi-layer feature fusion, and reconstruction. The RFDN is effectively derived from the residual feature distillation network(RFDB). We believe that the RFDB module is similar to the depth-wise separable convolution [14] in that it operates pointwise convolution and depth-wise convolution separately, using 3×3 convolution of residuals for intra-channel information interaction and 1×1 convolution for inter-channel information interaction, respectively. We think the part of depth-wise can be replaced by the module of pixel mixing (PM) without parameters to make it more efficient. In addition, BN layer [43]is added to make it more generalizable. An efficient feature distillation network (EFDN) is proposed as shown in Fig. 40. Specifically, in the first stage, shallow feature extraction consists of linear mapping and depth-wise convolution from the input image space to a higher dimensional feature space. Then stacked efficient pixel mixing Blocks (EPMB)(Fig. 41) are constructed for deep feature extraction, progressively refining the extracted features. The features generated by each EPMB are fused at the end of the backbone along the channel dimension.

Pixel Mixing Block. Pooling preserves the main features while reducing the number of parameters and computa-

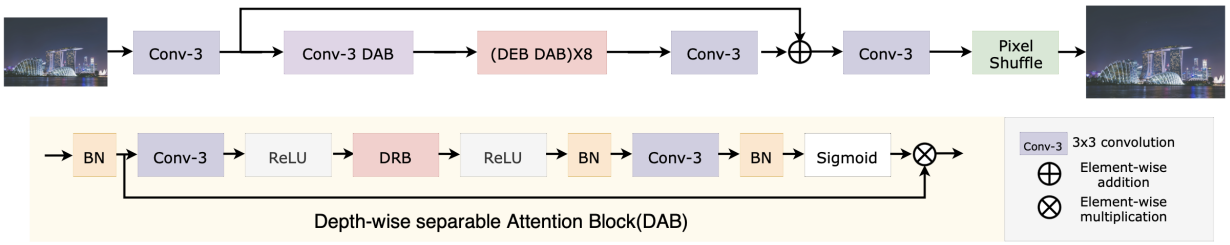


Figure 39. *Team LVGroup_HFUT:* The structure of EDRN.

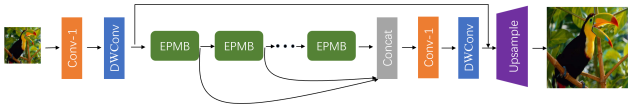


Figure 40. *FRL Team 4:* The architecture of efficient feature distillation network (EFDN)

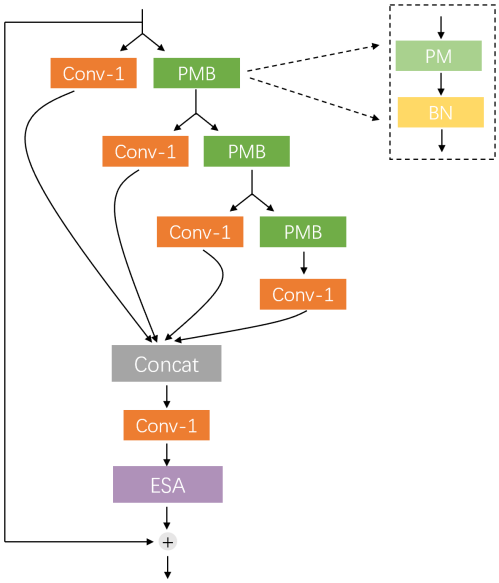


Figure 41. *FRL Team 4:* The architecture of the proposed efficient pixel mixing blocks (EPMB).

tions [31]. In the process of information filtering, Pooling tends to retain the high-level semantic features while losing some of them. To avoid the loss of low-level visual features while reducing parameters and computation, and to enhance the network generalization performance, we propose the module PM, which also has no parameters. PM divides the input into five groups equally, and moves the edge features to the opposite position in the order of left, right, top, and bottom for each of the first four groups, and keeps a copy of the features mixed with the relatively moved features. We describe this process as pixel Mixing.

Training Details Training was performed on DIV2K [1] and Flickr2K [65] images. HR patches of size 256 × 256 were randomly cropped from the HR images and the mini-batch size was set to 128. The model was trained with the ADAM optimizer, where $\beta_1 = 0.9$ and $\beta_2 = 0.9999$. The initial learning rate was set to 5×10^{-4} with cosine learning rate decay. The L2 loss was used for ab initio training and the number of iterations The model was implemented using Pytorch 1.10.1 and trained on 2 GeForce RTX 3090 GPUs.

4.26. Dase-IDEALab

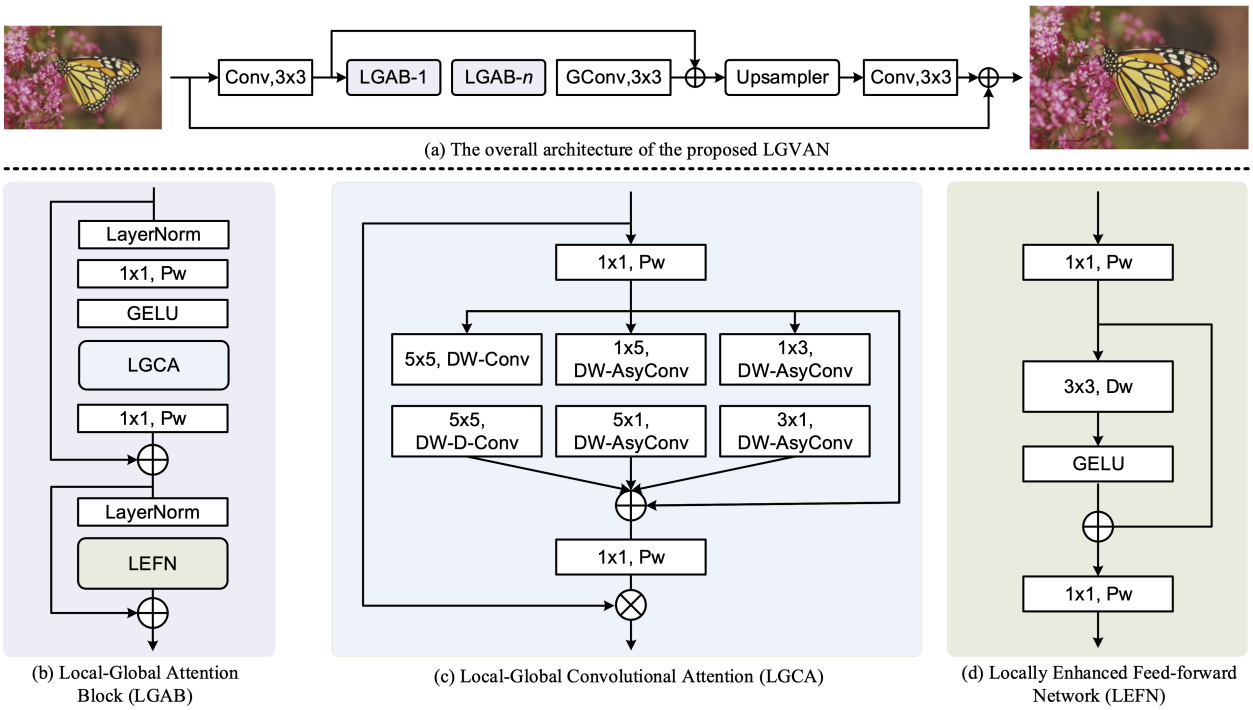
General method description. The Dase-IDEALab team proposed Global Visual Attention Network (LGVAN). The overall network structure is shown in Fig. 42(a). Following previous works, the network consists of three main parts: shallow feature extraction, deep feature extraction, and up-scaling reconstruction.

Inspired by VAN [32], the team proposed a Local-Global Attention Block (LGAB) tailored for efficient SR. As shown in Fig. 42(b), LGAB consists of Local-Global Convolutional Attention (LGCA), and Locally Enhanced Feed-forward Network (LEFN). LGCA takes a multi-branch structure to extract local and global features, which can be expressed as:

$$Attention = Conv_p \sum_{i=0}^{3} (Branch_i(Conv_p(F))) \quad (13)$$

$$Output = Attention \otimes F \quad (14)$$

where F denotes the input feature, while $Attention$ and $Output$ indicate attention map and the output feature respectively. $\otimes$ is element-wise multiplication. $Conv_p$ represents 1×1 point-wise convolution, and $Branch_i, i \in \{0, 1, 2, 3\}$ denotes i-th branch as shown in Fig. 42(c). $Branch_0$ consists of a 5×5 depth-wise convolution and a 5×5 depth-wise dilation convolution with dilation of 3 to capture long-range dependencies. $Branch_1$ and $Branch_2$ respectively consist of paired depth-wise asymmetric convolutions with kernel sizes of 3 and 5, which are used to extract local and texture information. Besides, $Branch_3$ is an identity mapping.



(a) The overall architecture of the proposed LGVAN

(b) Local-Global Attention Block (LGAB)

(c) Local-Global Convolutional Attention (LGCA)

(d) Locally Enhanced Feed-forward Network (LEFN)

Figure 42. *Team Dase-IDEALab:* The overall pipeline of Local-Global Visual Attention Network (LGVAN).

As shown in Fig. 42(d), a 3×3 depth-wise convolution is incorporated to strengthen LEFN, whilst the self-residual strategy [80] is introduced to overcome the drawbacks associated with depth-wise convolution. Mathematically, the LEFN can be formulated as:

$$F_e = Conv_e(F) \tag{15}$$

$$Output = Conv_c(\phi((Conv_d(F_e)) + F_e)) \tag{16}$$

where F denotes the input feature. $Conv_e$ and $Conv_c$ are two 1×1 convolutions carrying out channel expansion and contraction respectively. ϕ indicates GELU activation [35].

Loss Function During training, the model is trained with two types of loss functions $\mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_f$. Pixel-wise $\mathcal{L}_1$ loss is used to ensure that the generated content is close to ground truth:

$$\mathcal{L}_1 = ||R - T||_1 \tag{17}$$

where R and T denote the resulting SR image and the target image respectively. In addition, $\mathcal{L}_f$ is the frequency reconstruction loss [12] that enforces high-frequency details:

$$\mathcal{L}_f = ||\mathcal{F}(R) - \mathcal{F}(T)||_1 \tag{18}$$

where $\mathcal{F}(\cdot)$ represents the 2D Fast Fourier Transform. The weighting factor $\lambda = 0.5$ is used in the experiments.

Training details The proposed network consists of 9 LGABs, and the number of channels was set to 32. DIV2K [1] and Filckr2K [83] datasets were used for training. Data augmentation strategies involved in experiments included horizontal and vertical flips, and random rotations of 90, 180, and 270 degrees. In terms of the model optimization, the Adam [47] optimizer was used with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The initial learning rate of LGVAN was set to 5×10^{-4} and reduced by half after 2×10^5 iterations. During the training process, the mini-batch and the input patch size of LGVAN were set to 128 and 64×64. The model was trained using a total of 1×10^6 iterations.

4.27. FRL Team 0

General method description. There is already a lot of work trying to reduce the complexity of self-attention (SA), including [10, 15, 49]. Our goal is to reduce the complexity of transformers for SR and maintain its performance, so we propose Local-Global Term Transformer (LGTT) for SR, not every group needs self-attention. The overall architecture is shown in Fig. 43. In order to establish long-term dependencies efficiently, a pair of SAs is reserved in each group as a Global-term Modeler and using the striped window mechanism [76], avoiding redundant operations. we improve BWSA by adding multi-window and head mechanisms to improve computational efficiency,i.e., multi-window and head SA (M-WHSA), as shown in Fig. 44. In the rest of the blocks, we propose efficient pixel mixer (PM) modules without computational cost

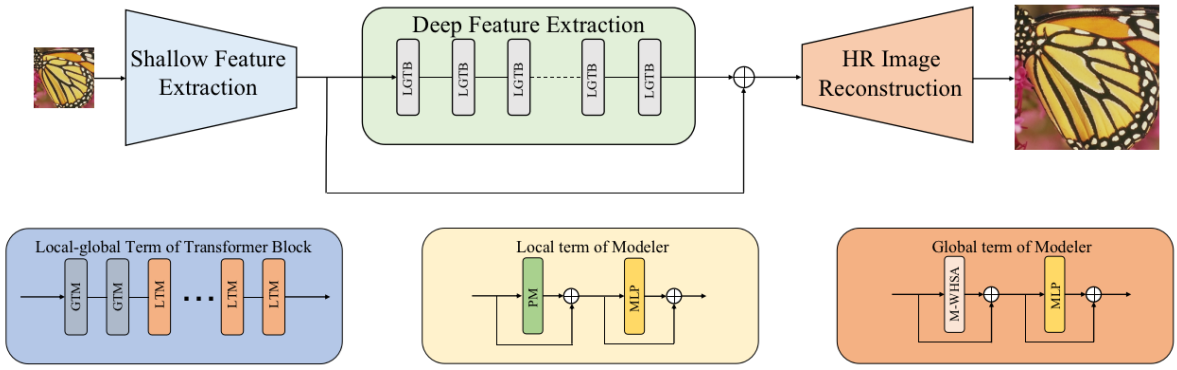


Figure 43. *FRL Team 0:* The architecture of efficient feature distillation network

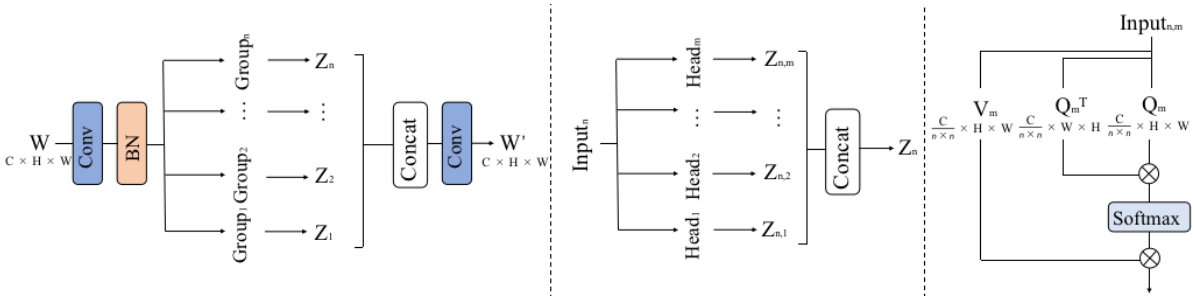


Figure 44. *FRL Team 0:* Illustration of our multi-window and heads SA mechanism based on [76]. n represents the number of headers and m represents the number of windows. The left figure shows the multi-window mechanism, which divides the groups according to the number of different windows of the input, as a way to increase the speed. The middle figure shows the multi-heads mechanism, where each group can have n heads in it while computing self-attention. The right figure shows the calculation of SA on a head.

and can efficiently model short-distance dependencies as a Local-term Modeler. which will be mentioned later about the PM module.

Pixel Mixer. We propose the pixel mixer module without parameters and computational complexity to create short-distance dependencies in each input Token. The PM module first divides the feature channels into five groups equally and moves the edge feature points to the opposite side in the order of left, right, top, and bottom for the first four groups. By inputting an intermediate feature of H × W × C, the PM module enables each input window in SA to obtain different local information according to other channels by linking the edge feature points with the opposite edge feature points respectively, so that each feature point can be fully utilized and the perceptual field of the latter module can be increased.

Training strategy. We use DF2K (DIV2K [1], Flickr2K [65])and LSDIR [59] for datasets. and propose that the channel input is set to 30, the data augmentation method with 90°, 180°, 270° random rotation and horizontal flip is used for training, the batch size is set to 128, and the input patch size of LR is 64×64.Trained using Adam optimizer [48] with β_1 = 0.9, β_2 = 0.999. The initialized learning rate is 5×10^{-4} and decays to 1×10^{-6} with the cosine learning rate. The model is optimized using the loss function of L_1 for a total of 1×10^6 iterations. Model training was performed using Pytorch [74] on two NVIDIA V100 32G GPUs.

4.28. FRL Team 3

Network Architecture. As shown in Fig. 45, HCAN contains three modules: (1) feature extraction, (2) nonlinear mapping, and (3) HR-image reconstruction. In the first stage, we use a 3×3 convolution layer to extract the coarse features from low-resolution images. Then we implement the nonlinear mapping by cascading multiple NTBs. Next, we use one 3×3 convolution layer and one nonparametric sub-pixel operation to reconstruct high-resolution images. Global shortcut connections are used. Our overall structure is neat, as complex topologies can lead to a serious reduction in inference speed [107].

Normalization-free Transformer Block. NTB uses a Transformer architecture and replaces self-attention (SA) with hierarchical context aggregation attention (HCAA) designed to capture short-range and long-range contexts at multiple levels. To further improve effectiveness and ef-

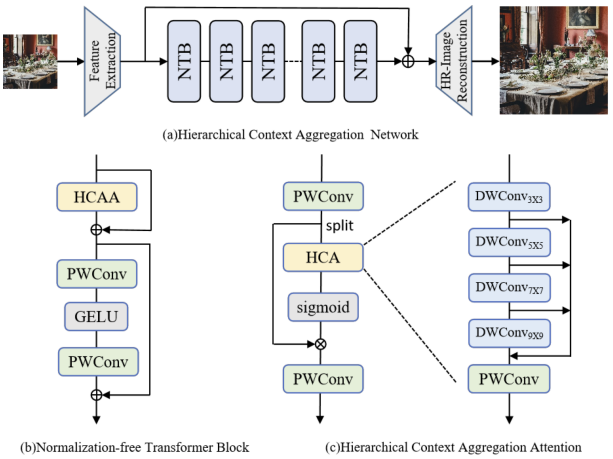


Figure 45. *FRL Team 3:* (a) is the overall pipeline of HCAN, which consists of cascading multiple NTBs (b). NTB uses a Transformer architecture without any normalization. Its core component is hierarchical context aggregation attention (HCAA), which can capture short-range and long-range contexts at multiple levels. Each DWconv is followed by a GELU [35].

ficiency, NTB removes Layer Normalization (LN) [3] from Transformer architecture and does not introduce other types of normalization. To the best of our knowledge, we are the first to use the Transformer architecture without any normalization in the super-resolution (SR) field. Normalization can speed up and stabilize training, but Layer Normalization (LN) increases inference time and memory usage, and Batch Normalization (BN) [44] damages the performance of SR [65]. We find that Sigmoid can replace the normalization effect to some extent without causing the above losses.

Hierarchical Context Aggregation Attention. Inspired by FocalNet [96], we replace SA with HCAA in Transformer architecture. The point-wise convolution (PWconv) is placed at the head and tail of the HCAA to exchange information between channels, The body part mainly acquires spatial contextual information through hierarchical depth-wise convolution (DWconv) with increasing kernel sizes in a cascading manner. The extracted features from each level of DWconv are added together to generate features with short-range and long-range contexts. The generated features are channel blended by PWconv and then attention maps are generated using Sigmoid. Finally, it is multiplied by elements with the unprocessed feature map. Compared to FocalNet, we remove the global aggregation and gated aggregation. The former impairs performance [46], while the latter impairs efficiency and leads to unstable training.

Training strategy. The proposed HCAN consists of 12 NTBs and the number of channels is set to 32. We train our models on DIV2K [1], Flickr2K [65] and LSDIR dataset [59]. Data augmentation methods of random rotation by 90°, 180°, 270° and flipping horizontally are utilized. The minibatch size is set to 128 and the patch size of each LR input is set to 64×64. The model is trained by L1 loss [93] with Adam optimizer [47] ($\beta_1 = 0.9, \beta_2 = 0.999$) for 1×10^6 total iterations. The learning rate is initialized as 5×10^{-4} and scheduled by cosine annealing learning.

4.29. AIIA-SR

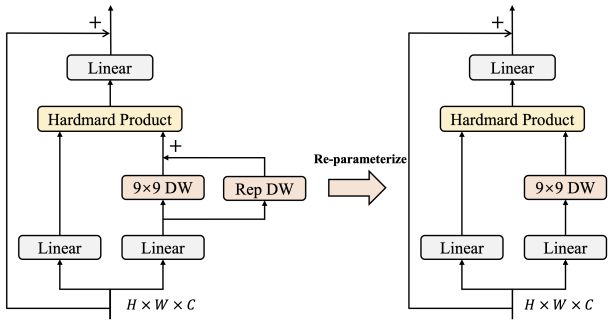


Figure 46. *Team AIIA-SR:* Illustration of the token mixer in ConvFormer Layer.

General method description. The AIIA-SR team presents an efficient image super-resolution network called CFSR, as illustrated in Fig. 47. CFSR is composed of three stages: shallow feature extraction, deep feature extraction, and image reconstruction. The shallow feature extraction module employs a 3×3 convolution to map the input RGB image into a latent feature space. The deep feature extraction module consists of two Residual ConvFormer Blocks (RCFB) and a 3×3 convolution layer. As depicted in Fig. 47(c) demonstrates that the CFL extends the MetaFormer [99] architecture, using the ConvFormer as the token mixer. Additionally, the team introduces a Mixed-Feed Forward Network, which enhances the feed-forward network by incorporating more positional prior information. Lastly, the reconstruction module adjusts the channel count using a 3×3 convolution and upsamples the latent feature to the target size via pixel-shuffle operation.

Building upon the work in [37], the AIIA-SR team introduces the ConvFormer Layer, which is an innovative attention-free transformer layer that utilizes large kernel depth-wise convolution. As depicted in Fig. 46, the team simplifies the traditional self-attention module by performing an element-wise product of the value and the convolutional feature generated by the large kernel depth-wise convolution. Furthermore, the team employs the re-parameterization strategy to enhance the stability of training and improve the overall performance of the model. Specifically, given the input feature X, this team computes the output feature Z using the following equations: $A = \mathrm{DConv}_{9\times9}(W_1X) + \mathrm{DConv}_{k'\times k'}(W_1X)$, $V = W_2X$,

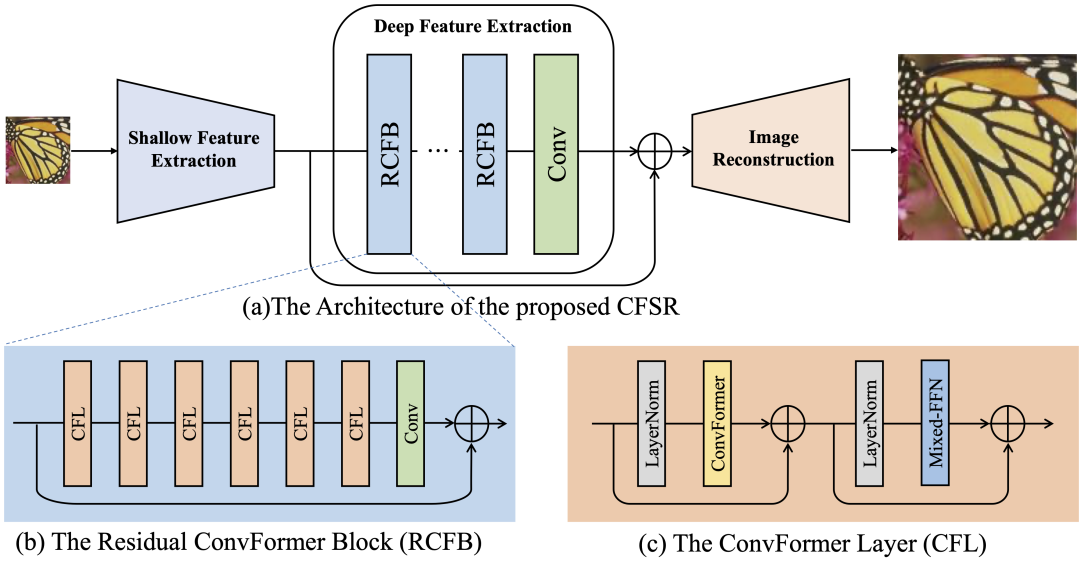


Figure 47. *Team AIIA-SR:* Detailed implementation and different components in the proposed CFSR.

$Z = W_3(A \odot V)$. Here, $\mathrm{DConv}_{k\times k}(\cdot)$ represents a depth-wise convolution with kernel size $k \times k$, W_1, W_2 and W_3 are weight matrices of three linear layers, and $\odot$ is the Hadamard product. Assuming that $k' < 9$, the calculation process of the convolutional feature A can be simplified as $A = \mathrm{DConv}_{9\times 9}(W_1 X)$, where $\mathrm{DConv}_{9\times 9}$ is obtained via re-parameterization [24] of trained $\mathrm{DConv}_{9\times 9}$ and $\mathrm{DConv}_{k'\times k'}$.

Mixed-FFN in CFL mixes a 3×3 depth-wise convolution and a MLP into each FFN, providing the positional information for transformer layers. It can be formulated as:

$$X'' = W''(\mathrm{GELU}(\mathrm{DConv}_{3\times 3}(W'X')) + X'), \quad (19)$$

where X'' is the output feature of Mixed-FFN, W' and W'' are weight matrices of two linear layers, and $\mathrm{GELU}(\cdot)$ is the activation function.

Training details. Based on the framework of BasicSR, this team uses DIV2K and Flickr2K as training sets to train in the upscaling factors of $\times 4$. During training, this team crops the image patches with the fixed size of 64×64 and sets the batch size to 16 for training. This team employs randomly rotating $90°$, $180°$, $270°$ and horizontal flip for data augmentation and uses ADAM with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to optimize L_1 loss. The initial learning rate is 2×10^{-4}. The CFSR is implemented by PyTorch and trained on two Nvidia RTX A4000 GPUs.

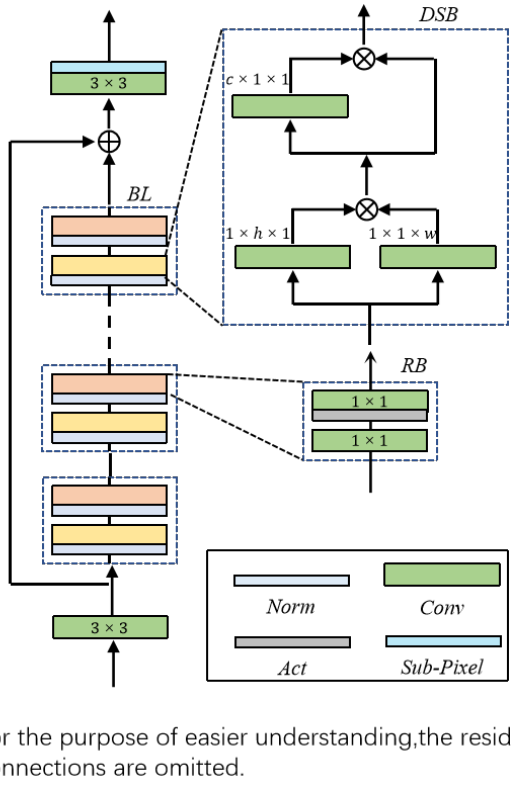


Figure 48. *FRL Team 2:* The structure SRneXt.

4.30. FRL Team 2

General method description. The structure is inspired by classical SR structures including CNNs and transformers which are commonly used in SR. Nowadays, transformers have been widely used in lightweight SR tasks. In ConvneXt [70], it is demonstrated that CNNs are not inferior to transformers in performance in many cases. Thus, the ad-

vantages of Transformer (like Swin transformer [64]) and CNN are analyzed.

Block layer. Block layers (BL) are the main layer in this structure. This layer consists of a residual block (RB) and a dimensional separable block (DSB).

Residual block. Residual blocks are used as the classical convolutional module in EDSR [65], where the BatchNorm is considered harmful and is removed. In the proposed network, LayerNorm is a residual block module to improve performance.

Dimensional separable block. Poolformer [100] confirmed that the advantage of transformers is their backbone structure and their self-attention can be replaced by convolution. Inspired by depthwise separable convolution [38], the dimensional separable block is proposed to divide the information features into height (H), width (W), and channel (C) dimensions. In Mobilenets [38], the feature information is divided point-wise (PW) and depth-wise (DW). In the proposed method, the H and W dimensions are processed together for intra-channel feature information and then the C dimension is processed for inter-channel feature information. Inspired by simple baselines for image restoration [8], the information of H and W is first multiplied to achieve non-linearity. After that, the output features are finally derived. Inspired by Inception-ResNet [82], the convolution kernel can be decomposed to balance network performance and the number of parameters. So group convolution is used for feature extraction in H and W dimensions.

Training strategy. Training was performed on DIV2K [1] and Flickr2K [65] images. HR patches of size 256×256 were randomly cropped from the HR images and the mini-batch size was set to 128. The model was trained with the ADAM optimizer [48], where $\beta_1 = 0.9$ and $\beta_2 = 0.9999$. The initial learning rate was set to 5×10^{-4} with cosine learning rate decay. The L2 loss was used for training. The model was implemented using Pytorch 1.10.1 and trained on 2 GeForce RTX 3090 GPUs.

4.31. CUIT_SRLab

General method description. Inspired by the Visual Attention Network [32] and VapSR [113], we propose a lightweight Symmetrical Visual Attention Network (SVAN) model, in which the large kernel convolution is decomposed by depth-wise convolution operation. SVAN ensures a large receptive field and achieves an efficient and lightweight attention structure. It generates better $\times$ 4 results with very few parameters.

As shown in Fig. 49, SVAN is divided into three parts: shallow feature extraction module, deep feature extraction module, and pixel-shuffle reconstruction module. The shallow feature map of input LR is generated by 3×3 convolution. Deep feature extraction consists of seven Symmetric Large Kernel Attention blocks (SLKAB), each of which is extended from 32 channels to 64 channels by 1×1 convolution and GELU activation to obtain more information. The deep features are obtained by two symmetric attention blocks, each composed of a 5×5 depth-wise convolution and a depth-wise dilation convolution with a kernel size of 5 and dilation of 3 and a 1×1 convolution. The convolution acceptance domain with kernel 17 can be obtained through the convolution combination, greatly reducing the number of computational parameters. Featured generated by the attention branch are fused with the original features using element-wise implementation. Another 1×1 convolution layer will reduce the number of channels to 32. Finally, pixel normalization is used to increase the stability in training. A 3×3 depth-wise dilation convolution layer with dilation of 3 follow SLKABs to further fuse the features. After the fusion of deep and shallow features, the upsampling of $\times4$ is realized through the reconstruction module of convolution and pixel-shuffle layers.

Training Details. During the training, HR patches of size 256×256 are randomly cropped from HR images. The mini-batch size is set to 196 and the number of feature channels is set to 64. The learning rate is set to 1×10^{-4} for 500 epochs during pre-training. The LSDIR [59] dataset, the Adam optimizer, and L1 loss function are used. After that, DIV2K [1] and Flickr2K [83] datasets are used for model fine-tuning. The optimizer and loss are unchanged. The initial learning rate is set to 5×10^{-5} and halved at every 200 epochs. After 3000 epochs, L2 loss is used for fine-tuning. And The initial learning rate is set to 1×10^{-4} and halved at every 200 epochs. The total number of epochs is 10000.

Acknowledgments

We thank the NTIRE 2023 sponsors: Sony Interactive Entertainment, Meta Reality Labs, ModelScope, ETH Zürich (Computer Vision Lab) and University of Würzburg (Computer Vision Lab).

A. Teams and affiliations

NTIRE 2023 team

Title: NTIRE 2023 Efficient Super-Resolution Challenge

Members:

Yawei Li[1] (yawei.li@vision.ee.ethz.ch),
Yulun Zhang[1] (yulzhang@ethz.ch),
Luc Van Gool[1] (vangool@vision.ee.ethz.ch),
Radu Timofte[1,2] (radu.timofte@uni-wuerzburg.de)

Affiliations:

[1] Computer Vision Lab, ETH Zurich, Switzerland
[2] Computer Vision Lab, University of Würzburg, Germany

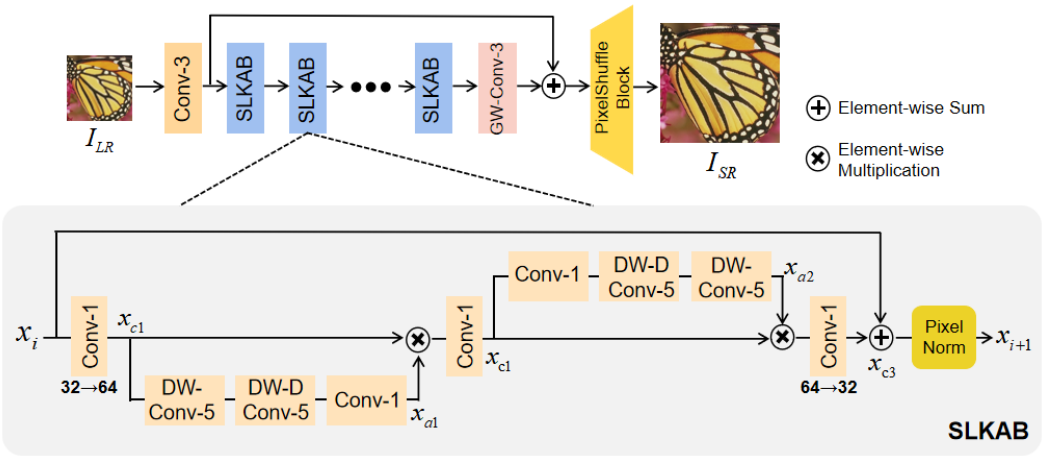


Figure 49. *Team CUIT_SRLab:* Symmetrical Visual Attention Network (SVAN) and Symmetrical Large Kernel Attention Block (SLKAB)

MegSR

Title: Efficiency Distillation and Iterative Pruning for Single Image Super-Resolution
Members:
Lei Yu[1] (yulei02@megvii.com), Xinpeng Li[1], Youwei Li[3], Qi Wu[1], Ting Jiang[1], Mingyan Han[1], Wenjie Lin[1], Chengzhi Jiang[1], Jinting Luo[1], Tong Peng[1], Haoqiang Fan[1] and Shuaicheng Liu[2,1*]
Affiliations:
[1] Megvii Technology
[2] University of Electronic Science and Technology of China (UESTC)
[3] MicroBT

Zapdos

Title: A Single Residual Network with ESA Modules and Distillation
Members:
Yucong Wang[1] (1401121556@qq.com), Minjie Cai[1]
Affiliations:
[1] College of Computer Science and Electronic Engineering, Hunan University

DFCDN

Title: Efficient Image Super-Resolution with Deep Feature Complement and Distillation Network
Members:
Mingxi Li[1] (li_mx_0318@163.com), Yuhang Zhang, Xianjun Fan[1], Yankai Sheng[1]
Affiliations:
[1] Attrsense

TelunXupt

Title: Multi-level Dispersion Residual Network for Efficient Image Super-Resolution
Members:
Yanyu Mao[1] (bolt35982@gmail.com), Nihao Zhang[1], Qian Wang[1,2]
Affiliations:
[1] Xian University of Posts and Telecommunications, Xi'an, China
[2] National Engineering Laboratory for Cyber Event Warning and Control Technologies

NJUST_E

Title: A Lightweight Multi-scale Feature Attention for Image Super-Resolution
Members:
Mingjun Zheng(1353613718@qq.com), Long Sun, Jinshan Pan, Jiangxin Dong, Jinhui Tang
Affiliations:
Nanjing University of Science and technology

NJUST_M

Title: Gated Feature Modulation for Efficient Super-Resolution
Members:
Zhongbao Yang(yangzhongbao40@gmail.com), Long Sun, Jinshan Pan, Jiangxin Dong, Jinhui Tang
Affiliations:
Nanjing University of Science and technology

KaiBai_Group

Title: Partial Feature Distillation Network for Efficient Super-Resolution
Members:
Yan Wang[1] (wyrmy@foxmail.com),
Erlin Pan[2],
Qixuan Cai[3],
Xinan Dai[3]
Affiliations:
[1] Nankai University
[2] University of Electronic Science and Technology of China
[3] Tianjin University

NoahTerminalCV_TeamA

Title: NoahESRNet: All You Need is to Optimize Attention
Members:
Magauiya Zhussip[1] (magauiya.zhussip1@huawei.com), Nikolay Kalyazin[1], Dmitry Vyal[1], Xueyi Zou[1], Youliang Yan[1]
Affiliations:
[1] Noah's Ark Lab, Huawei Technologies

NoahTerminalCV_TeamB

Title: MobileNoahESRNet: More Kernel Decomposition, Less Parameters
Members:
Dmitry Vyal[1] (vyal.dmitry@huawei.com), Magauiya Zhussip[1], Nikolay Kalyazin[1], Xueyi Zou[1], Youliang Yan[1]
Affiliations:
[1] Noah's Ark Lab, Huawei Technologies

SeaOuter

Title: Low Memory Residual Local Feature Network for Efficient Super-Resolution
Members:
Heaseo Chung[1,2] (heaseochung@gmail.com)
Affiliations:
[1] McMaster University, Canada
[2] Espresomedia, South Korea

Antins_cv

Title: Solution for NTIRE 2023 Efficient SR Challenge
Members:
Jin Zhang[1] (zj346862@antgroup.com), Gaocheng Yu[2], Feng Zhang[3], Hongbin Wang[4]
Affiliations:
[1] AntGroup

Young

Title: Channel-Aware Re-parameterization Network
Members:
Bohao Liao[1] (liaobh@mail.ustc.edu.cn), Zhibo Du[1], Yuliang Wu[1], Gege Shi[1], Long Peng[1], Yang Wang[1], Yang Cao[1], Zhengjun Zha[1]

Affiliations:
[1] University of Science and Technology of China

NTU607_ESR

Title: Finetuning and pruning for efficient super-resolution
Members:
Zhi-Kai Huang[2] (brent5481@gmail.com), Yi-Chung Chen[3], Yuan-Chun Chiang[2], Hao-Hsiang Yang[2],
Wei-Ting Chen[1], Hua-En Chang[2], I-Hsiang Chen[2], Chia-Hsuan Hsieh[4], Sy-Yen Kuo[2]
Affiliations:
[1] Graduate Institute of Electronics Engineering, National Taiwan University, Taiwan
[2] Department of Electrical Engineering, National Taiwan University, Taiwan
[3] Graduate Institute of Communication Engineering, National Taiwan University, Taiwan
[4] ServiceNow, USA

CMVG

Title: Knowledge Distillation and Re-parameterization for Efficient Image Super-Resolution
Members:
Xin Liu[1] (liuxin9976@163.com), Qian Wang[1], Jiahao Pan[2]
Affiliations:
[1] China Mobile Research Institute
[2] Chongqing University of Technology

Touch_Fish

Title: Lightening Attention: Rethinking the Attention Mechanism in Efficient Image Enhancement
Members:
Hongyuan Yu[1] (yuhongyuan@xiaomi.com), Weichen Yu[2], Lin Ge[1], Jiahua Dong[3], Yajun Zou[1], Zhuoyuan Wu[1], Binnan Han[1], Xiaolin Zhang[1], Heng Zhang[1], Xuanwu Yin[1], Kunlong Zuo[1]
Affiliations:
[1] Multimedia Department, Xiaomi Inc.
[2] Institute of Automation, Chinese Academy of Sciences

[3] Shenyang Institute of Automation, Chinese Academy of Sciences

CUC_SR

Title: Reparameterized Residual Feature Network For Lightweight Image Super-Resolution
Members:
Weijian Deng[1] (348957269@qq.com), Hongjie Yuan[1], Zengtong Lu[2]
Affiliations:
[1] School of Information and Communication Engineering, Communication University of China.
[2] Smart Classroom Division, Ruijie Networks Co., Ltd.

NJUST_R

Title: Spatially-Adaptive Feature Modulation for Efficient Image Super-Resolution
Members:
Long Sun(cs.longsun@gmail.com), Zhongbao Yang, Jinshan Pan, Jiangxin Dong, Jinhui Tang
Affiliations:
Nanjing University of Science and technology

Sissie_Lab

Title: Diversified Local Feature Arch-Network for Efficient Super-Resolution
Members:
Mingyu Ouyang[1] (yyyang404@whu.edu.cn), Wenzhuo Ma[1], Nian Liu[1], Hanyou Zheng[1], Yuantong Zhang[1], Junxi Zhang[1], Zhenzhong Chen[1]
Affiliations:
[1] School of Remote Sensing and Information Engineering, Wuhan University
[2] School of Computer Science, Wuhan University

GarasSjtu

Title: Mixer-based Local Residual Network for Lightweight Image Super-resolution
Members:
Garas Gendy[1] (garasgaras@yahoo.com),
Nabil Sabor[2] ,
Jingchao Hou[1] ,
Guanghui He[1]
Affiliations:
[1] Micro-Nano Electronics Department, Shanghai Jiao Tong University, Shanghai 200240, China.
[2] Electrical Engineering Department, Faculty of Engineering, Assiut University, Assiut 71516, Egypt

USTC_ESR

Title: ***Enhanced Fast Residual Network for Efficient Image Super Resolution***
Members:
Yurui Zhu [1] (zyr@mail.ustc.edu.cn), Xi Wang[1], Xueyang Fu[1], Zheng-Jun Zha [1]
Affiliations:
[1] University of Science and Technology of China

SEU_CNII

Title: Expand Big, Then Shrink: An Automatic Way to Generating Efficient Super Resolution Model
Members:
Daheng Yin[1] (yindaheng98@seu.edu.cn), Mengyang Liu[1], Baijun Chen[1]
Affiliations:
[1] Southeast University

AVC2_CMHI_SR

Title: Residual Interactive Partial Feature Distillation Network for Lightweight Image Super-resolution
Members:
Ao Li[1] (aoli@std.uestc.edu.cn), Lei Luo[2], Kangjun Jin[3], Ce Zhu[1]
Affiliations:
[1] University of Electronic Science and Technology of China, Chengdu, China.
[2] Chongqing University of Posts and Telecommunications, Chongqing, China.
[3] China Mobile (Hangzhou) Information Technology Co., Ltd, Hangzhou, China.

Set5_Baby

Title: Large Kernel Distillation Network for Efficient Single Image Super-Resolution
Members:
Xiaoming Zhang[1] (xiaoming.zhang@my.swjtu.edu.cn), Chengxing Xie[1], Linze Li[1], Haiteng Meng[1], Tianlin Zhang[2], Tianrui Li[1], Xiaole Zhao[1]
Affiliations:
[1] Southwest Jiaotong University, China
[2] National Space Science Center, Chinese Academy of Science, China

LVGroup_HFUT

Title: Efficient Deep Residual Network for Efficient Super-Resolution
Members:

Zhao Zhang[1] (cszzhang@gmail.com),
Baiang Li[1],
Huan Zheng[1]
Suiyi Zhao[1]
Yangcheng Gao[1]
Jiahuan Ren[1]
Affiliations:
[1] Hefei University of Technology, Hefei, China

FRL Team 4

Title: Efficient feature distillation network for Image Super-Resolution
Members:
Kang Hu[1] (hukangyy@gmail.com)
Jingpeng Shi[2],
Affiliations:
[1] Anhui University,China
[2] Fried Rice Lab

Dase-IDEALab

Title: Local-Global Visual Attention Network
Members:
Zhijian Wu[1] (zjwu_97@stu.ecnu.edu.cn),
Dingjiang Huang[1]
Affiliations:
[1] School of Data Science and Engineering, East China Normal University

FRL Team 0

Title: LGTT: Local-Global Term Transformer for SR
Members:
Jinchen Zhu[1] (jinchen.Z@outlook.com), Jinpeng Shi[1], Hui Li[2], Qianru Xv[2]
Affiliations:
[1] School of Electronic Information Engineering,Anhui University,China
[2] Fried Rice Lab

FRL Team 3

Title: Hierarchical Context Aggregation Network
Members:
Tianle Liu [1,2] (tianle.l@outlook.com),
Jinpeng Shi [1,2] ,
Shizhuang Weng [1]
Affiliations:
[1] School of Electronic Information Engineering, Anhui University
[2] Fried Rice Lab

AIIA-SR

Title: CFSR: A Simple Hierarchical Baseline for Lightweight Image Super-Resolution
Members:
Gang Wu[1] (gwu@hit.edu.cn), Junpeng Jiang[1], Xianming Liu[1], Junjun Jiang[1]
Affiliations:
[1] Harbin Institute of Technology

FRL Team 2

Title: SRneXt:Improvement on ConvneXt
Members:
Mingjian Zhang[1] (zhang9317112@gmail.com), Shizhuang Weng[1], Jinpeng Shi[2]
Affiliations:
[1] School of Electronic Information Engineering, Anhui Universit
[2] Fried Rice Lab

CUIT_SRLab

Title: Symmetrical Visual Attention Network for Efficient Image Super-Resolution
Members:
Jing Hu[1] (jing_hu09@163.com), Chengxu Wu[1], Qinrui Fan[1], Chengming Feng[1], Ziwei Luo[2], Shu Hu[3], Siwei Lyu[4], Xi Wu[1], Xin Wang[4]
Affiliations:
[1] Department of Computer Science, Chengdu University of Information Technology, Chengdu, China
[2] Uppsala University, Sweden
[3] Heinz College of Information Systems and Public Policy, Carnegie Mellon University, USA
[4] Department of Computer Science and Engineering at the University at Buffalo, State University of New York, USA

R.I.P ShopeeVideo

Title: R.I.P ShopeeVideo
Members:
Chengpeng Chen[1] (chencp@live.com),

USTC_ESR

Title: Enhanced Fast Residual Network for Efficient Image Super Resolution
Members:
Yurui Zhu [1] (zyr@mail.ustc.edu.cn), Xi Wang[1], Xueyang Fu[1], Zheng-Jun Zha [1]
Affiliations:
[1] University of Science and Technology of China

FRL Team 1

Title: FRL Team 1
Members:
Yulong Liu[1] (yl.liu88@outlook.com), Jinpeng Shi[1]
Affiliations:
[1] Anhui University, Hefei, China

Loading2

Title: Loading2
Members:
Min Gao[1] (1157632500@qq.com), Jingwen Zhang[1], Ruonan Wang[1]
Affiliations:
[1] Xidian University

Alpha

Title: Alpha
Members:
Jianbin Zheng[1] (2327396173@qq.com)
Affiliations:
[1] South China University of Technology, Guangzhou, Guangdong, China

References

[1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017. 2, 4, 6, 8, 10, 12, 13, 14, 17, 19, 22, 24, 25, 26, 27, 28, 30

[2] Codruta O Ancuti, Cosmin Ancuti, Florin-Alexandru Vasluianu, Radu Timofte, et al. NTIRE 2023 challenge on nonhomogeneous dehazing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 28

[4] Kartikeya Bhardwaj, Milos Milosavljevic, Liam O'Neil, Dibakar Gope, Ramon Matas, Alex Chalfin, Naveen Suda, Lingchuan Meng, and Danny Loh. Collapsible linear blocks for super-efficient super resolution. *arXiv preprint arXiv:2103.09404*, 2021. 11

[5] Mingdeng Cao, Chong Mou, Fanghua Yu, Xintao Wang, Yinqiang Zheng, Jian Zhang, Chao Dong, Ying Shan, Gen Li, Radu Timofte, et al. NTIRE 2023 challenge on 360° omnidirectional image and video super-resolution: Datasets, methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[6] Du Chen, Jie Liang, Xindong Zhang, Ming Liu, Hui Zeng, and Lei Zhang. Human guided ground-truth generation for realistic image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2023. 5

[7] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. Run, don't walk: Chasing higher flops for faster neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023. 21, 22

[8] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *Proceedings of European Conference on Computer Vision*, pages 17–33. Springer, 2022. 30

[9] Xiangyu Chen, Xintao Wang, Jiantao Zhou, and Chao Dong. Activating more pixels in image super-resolution transformer. *arXiv preprint arXiv:2205.04437*, 2022. 5

[10] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 26

[11] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. In *ICCV*, 2021. 10, 20

[12] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. In *ICCV*, pages 4641–4650, 2021. 26

[13] Haram Choi, Jeongmin Lee, and Jihoon Yang. N-gram in swin transformers for efficient lightweight image super-resolution. *arXiv preprint arXiv:2211.11436*, 2022. 18

[14] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 24

[15] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020. 26

[16] Marcos V Conde, Manuel Kolmet, Tim Seizinger, Thomas E. Bishop, Radu Timofte, et al. Lens-to-lens bokeh effect transformation. NTIRE 2023 challenge report. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[17] Marcos V Conde, Eduard Zamfir, Radu Timofte, et al. Efficient deep models for real-time 4k image super-resolution. NTIRE 2023 benchmark and report. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2, 3, 5

[18] Weijian Deng, Hongjie Yuan, Lunhui Deng, and Zengtong Lu. Reparameterized residual feature network for lightweight image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 18

[19] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1911–1920, 2019. 19

[20] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse branch block: Building a convolution as an inception-like unit. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10886–10895, 2021. 19

[21] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Diverse branch block: Building a convolution as an inception-like unit. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10881–10890, 2021. 20

[22] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021. 17

[23] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *CVPR*, 2021. 20

[24] Xiaohan Ding, Xiangyu Zhang, Yizhuang Zhou, Jungong Han, Guiguang Ding, and Jian Sun. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2022. 29

[25] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *Proceeding of the European Conference on Computer Vision*, pages 184–199. Springer, 2014. 1

[26] Zongcai Du, Ding Liu, Jie Liu, Jie Tang, Gangshan Wu, and Lean Fu. Fast and memory-efficient network towards efficient image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 853–862, June 2022. 11, 12, 13, 14, 24

[27] Zongcai Du, Jie Liu, Jie Tang, and Gangshan Wu. Anchor-based plain net for mobile image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2494–2502, 2021. 21

[28] Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. *The Thirty-Fourth IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 22

[29] Garas Gendy, Nabil Sabor, Jingchao Hou, and Guanghui He. Mixer-based local residual network for lightweight image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop*, 2023. 20

[30] Garas Gendy, Nabil Sabor, Jingchao Hou, and Guanghui He. Real-time channel mixing net for mobile image super-resolution. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 573–590. Springer, 2023. 21

[31] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020. 25

[32] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022. 25, 30

[33] Daniel Haase and Manuel Amthor. Rethinking depthwise separable convolutions: How intra-kernel correlations lead to improved mobilenets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14600–14609, 2020. 23

[34] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1580–1589, 2020. 7, 10

[35] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 21, 23, 24, 26, 28

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[37] Qibin Hou, Chengze Lu, Ming-Ming Cheng, and Jiashi Feng. Conv2former: A simple transformer-style convnet for visual recognition. *CoRR*, abs/2211.11943, 2022. 28

[38] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 30

[39] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018. 14

[40] Mu Hu, Junyi Feng, Jiashen Hua, Baisheng Lai, Jianqiang Huang, Xiaojin Gong, and Xian-Sheng Hua. Online convolutional re-parameterization. *CoRR*, abs/2204.00826, 2022. 7

[41] Huaibo Huang, Ran He, Zhenan Sun, and Tieniu Tan. Wavelet-srnet: A wavelet-based cnn for multi-scale face super resolution. In *Proceedings of the IEEE international conference on computer vision*, pages 1689–1697, 2017. 13

[42] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *Proceedings of the ACM International Conference on Multimedia*, pages 2024–2032, 2019. 8, 11, 12, 23

[43] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 24

[44] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 28

[45] Xiaoyang Kang, Xianhui Lin, Kai Zhang, Zheng Hui, Wangmeng Xiang, Jun-Yan He, Xiaoming Li, Peiran Ren, Xuansong Xie, Radu Timofte, et al. NTIRE 2023 video colorization challenge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[46] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1637–1645, 2016. 28

[47] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 12, 18, 26, 28

[48] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 27, 30

[49] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020. 26

[50] Fangyuan Kong, Mingxi Li, Songwei Liu, Ding Liu, Jingwen He, Yang Bai, Fangmin Chen, and Lean Fu. Residual local feature network for efficient super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 766–776, 2022. 5, 14, 21, 22

[51] Fangyuan Kong, Mingxi Li, Songwei Liu, Ding Liu, Jingwen He, Yang Bai, Fangmin Chen, and Lean Fu. Residual local feature network for efficient super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 766–776, June 2022. 6, 7, 12, 13, 15, 16, 20

[52] Fangyuan Kong, Mingxi Li, Songwei Liu, Ding Liu, Jingwen He, Yang Bai, Fangmin Chen, and Lean Fu. Residual local feature network for efficient super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 766–776, 2022. 10, 11, 12, 13, 18, 19, 20, 21

[53] Wonkyung Lee, Junghyup Lee, Dohyung Kim, and Bumsub Ham. Learning with privileged information for efficient image super-resolution. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 465–482. Springer, 2020. 15, 17

[54] Yawei Li, Eirikur Agustsson, Shuhang Gu, Radu Timofte, and Luc Van Gool. CARN: convolutional anchored regression network for fast and accurate single image super-resolution. In *Proceeding of the European Conference on Computer Vision Workshops*, pages 166–181. Springer, 2018. 1

[55] Yawei Li, Yuchen Fan, Xiaoyu Xiang, Denis Demandolx, Rakesh Ranjan, Radu Timofte, and Luc Van Gool. Efficient and explicit modelling of image hierarchies for image restoration. *arXiv preprint arXiv:2303.00748*, 2023. 1

[56] Yawei Li, Shuhang Gu, Luc Van Gool, and Radu Timofte. Learning filter basis for convolutional neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5623–5632, 2019. 2

[57] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. DHP: Differentiable meta pruning via hypernetworks. In *Proceeding of the European Conference on Computer Vision*, pages 608–624. Springer, 2020. 2

[58] Yawei Li, Wen Li, Martin Danelljan, Kai Zhang, Shuhang Gu, Luc Van Gool, and Radu Timofte. The heterogeneity hypothesis: Finding layer-wise differentiated network architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2144–2153, 2021. 2

[59] Yawei Li, Kai Zhang, Jingyun Liang, Jiezhang Cao, Ce Liu, Rui Gong, Yulun Zhang, Hao Tang, Yun Liu, Denis Demandolx, et al. Lsdir: A large scale dataset for image restoration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2, 4, 5, 6, 8, 10, 13, 14, 17, 20, 21, 22, 27, 28, 30

[60] Yawei Li, Kai Zhang, Radu Timofte, Luc Van Gool, et al. NTIRE 2022 challenge on efficient super-resolution: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2022. 3, 11, 12, 13, 14

[61] Yawei Li, Yulun Zhang, Luc Van Gool, Radu Timofte, et al. NTIRE 2023 challenge on image denoising: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[62] Zheyuan Li, Yingqi Liu, Xiangyu Chen, Haoming Cai, Jinjin Gu, Yu Qiao, and Chao Dong. Blueprint separable residual network for efficient image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 833–843, 2022. 8, 23, 24

[63] Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5958–5968. PMLR, 2020. 22

[64] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021. 30

[65] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1132–1140, 2017. 1, 6, 7, 8, 10, 24, 25, 27, 28, 30

[66] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *Proceedings of the European Conference on Computer Vision Workshops*, pages 41–55. Springer, 2020. 2, 3, 8, 11, 12, 18, 19, 22, 24

[67] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 41–55. Springer, 2020. 13, 22

[68] Jie Liu, Wenjie Zhang, Yuting Tang, Jie Tang, and Gangshan Wu. Residual feature aggregation network for image super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2359–2368, 2020. 8, 20, 21, 23

[69] Xiaohong Liu, Xiongkuo Min, Wei Sun, Yulun Zhang, Kai Zhang, Radu Timofte, Guangtao Zhai, Yixuan Gao,

Yuqin Cao, Tengchuan Kou, Yunlong Dong, Ziheng Jia, et al. NTIRE 2023 quality assessment of video enhancement challenge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[70] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022. 29

[71] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. MetaPruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 2

[72] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 10, 20

[73] Yanyu Mao, Nihao Zhang, Qian Wang, Bendu Bai, et al. Multi-level dispersion residual network for efficient image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 8

[74] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 27

[75] Woo Sanghyun, Debnath Shoubhik, Hu Ronghang, Chen Xinlei, Liu Zhuang, Kweon In So, and Xie Saining. Convnext v2: Co-designing and scaling convnets with masked autoencoders. *arXiv preprint arXiv:2301.00808*, 2023. 20

[76] Jinpeng Shi, Hui Li, Tianle Liu, Yulong Liu, Mingjian Zhang, Jinchen Zhu, Ling Zheng, and Shizhuang Weng. Image super-resolution using efficient striped window transformer. *arXiv preprint arXiv:2301.09869*, 2023. 26, 27

[77] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 9, 10, 20

[78] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016. 24

[79] Alina Shutova, Egor Ershov, Georgy Perevozchikov, Ivan A Ermakov, Nikola Banic, Radu Timofte, Richard Collins, Maria Efimova, Arseniy Terekhin, et al. NTIRE 2023 challenge on night photography rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[80] Bin Sun, Yulun Zhang, Songyao Jiang, and Yun Fu. Hybrid pixel-unshuffled network for lightweight image super-resolution. *arXiv preprint arXiv:2203.08921*, 2022. 26

[81] Long Sun, Jiangxin Dong, Jinhui Tang, and Jinshan Pan. Spatially-adaptive feature modulation for efficient image super-resolution. *arXiv preprint arXiv:2302.13800*, 2023. 10, 19

[82] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017. 30

[83] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPR workshops*, pages 114–125, 2017. 12, 26, 30

[84] Asher Trockman and J Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022. 20

[85] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966. 12

[86] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1365–1374, 2019. 2

[87] Florin-Alexandru Vasluianu, Tim Seizinger, Radu Timofte, et al. NTIRE 2023 image shadow removal challenge report. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[88] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022. 14

[89] Longguang Wang, Yulan Guo, Yingqian Wang, Juncheng Li, Shuhang Gu, Radu Timofte, et al. NTIRE 2023 challenge on stereo image super-resolution: Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[90] Yan Wang. Edge-enhanced feature distillation network for efficient super-resolution. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, pages 777–785, 2022. 10

[91] Yucong Wang and Minjie Cai. A single residual network with esa modules and distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 6

[92] Yingqian Wang, Longguang Wang, Zhengyu Liang, Jungang Yang, Radu Timofte, Yulan Guo, et al. NTIRE 2023 challenge on light field image super-resolution: Dataset, methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[93] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 28

[94] Chengxing Xie, Xiaoming Zhang, Linze Li, Haiteng Meng, Tianlin Zhang, Tianrui Li, and Xiaole Zhao. Large kernel distillation network for efficient single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 23

[95] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum al-

gorithm for faster optimizing deep models. *arXiv preprint arXiv:2208.06677*, 2022. 24

[96] Jianwei Yang, Chunyuan Li, and Jianfeng Gao. Focal modulation networks. *arXiv preprint arXiv:2203.11926*, 2022. 28

[97] Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. Reducing bert pre-training time from 3 days to 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019. 12

[98] Lei Yu, Xinpeng Li, Youwei Li, Ting Jiang, Qi Wu, Haoqiang Fan, and Shuaicheng Liu. Dipnet: Efficiency distillation and iterative pruning for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 5

[99] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10819–10829, 2022. 28

[100] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 10819–10829, 2022. 30

[101] Pierluigi Zama Ramirez, Fabio Tosi, Luigi Di Stefano, Radu Timofte, et al. NTIRE 2023 challenge on hr depth from images of specular and transparent surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[102] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022. 10

[103] Dafeng Zhang, Feiyu Huang, Shizhuo Liu, Xiaobing Wang, and Zhezhu Jin. Swinfir: Revisiting the swinir with fast fourier convolution and improved training for image super-resolution, 2022. 14

[104] Kai Zhang, Martin Danelljan, Yawei Li, Radu Timofte, Jie Liu, Jie Tang, Gangshan Wu, Yu Zhu, Xiangyu He, Wenjie Xu, et al. AIM 2020 challenge on efficient super-resolution: Methods and results. In *Proceedings of the European Conference on Computer Vision Workshops*, pages 5–40. Springer, 2020. 2

[105] Linfeng Zhang, Xin Chen, Xiaobing Tu, Pengfei Wan, Ning Xu, and Kaisheng Ma. Wavelet knowledge distillation: Towards efficient image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12464–12474, 2022. 13

[106] Xindong Zhang, Huiyu Zeng, and Lei Zhang. Edge-oriented convolution block for real-time super resolution on mobile devices. *Proceedings of the 29th ACM International Conference on Multimedia*, 2021. 7, 11, 12, 20

[107] Xindong Zhang, Hui Zeng, and Lei Zhang. Edge-oriented convolution block for real-time super resolution on mobile devices. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4034–4043, 2021. 16, 18, 19, 27

[108] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2015. 2

[109] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 286–301, 2018. 8

[110] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1

[111] Yulun Zhang, Kai Zhang, Zheng Chen, Yawei Li, Radu Timofte, et al. NTIRE 2023 challenge on image super-resolution (x4): Methods and results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023. 2

[112] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016. 19

[113] Lin Zhou, Haoming Cai, Jinjin Gu, Zheyuan Li, Yingqi Liu, Xiangyu Chen, Yu Qiao, and Chao Dong. Efficient image super-resolution using vast-receptive-field attention. In *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II*, pages 256–272. Springer, 2023. 30

[114] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017. 5

[115] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *Proceedings of International Conference on Learning Representations*, 2017. 2

[116] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018. 2