Spatially-Adaptive Feature Modulation for Efficient Image Super-Resolution

Long Sun, Jiangxin Dong, Jinhui Tang, and Jinshan Pan*
School of Computer Science and Engineering, Nanjing University of Science and Technology

Abstract

Although deep learning-based solutions have achieved impressive reconstruction performance in image super-resolution (SR), these models are generally large, with complex architectures, making them incompatible with low-power devices with many computational and memory constraints. To overcome these challenges, we propose a spatially-adaptive feature modulation (SAFM) mechanism for efficient SR design. In detail, the SAFM layer uses independent computations to learn multi-scale feature representations and aggregates these features for dynamic spatial modulation. As the SAFM prioritizes exploiting non-local feature dependencies, we further introduce a convolutional channel mixer (CCM) to encode local contextual information and mix channels simultaneously. Extensive experimental results show that the proposed method is ***3***× *smaller than state-of-the-art efficient SR methods, e.g., IMDN, and yields comparable performance with much less memory usage. Our source codes and pre-trained models are available at:* `https://github.com/sunny2109/SAFMN`.

1. Introduction

Single image super-resolution (SISR) aims to restore a high-resolution (HR) image from its low-resolution (LR) counterpart by recovering lost details. This longstanding and challenging task has recently attracted much attention due to the rapid development of streaming media or high-definition devices. As these scenarios are usually resource-constrained, it is of great interest to develop an efficient and effective SR method to estimate HR images for better visual display on these platforms or products.

Deep learning-based SR methods have achieved significant performance improvements with the great evolution of hardware technologies, as we can use large amounts of data to train much larger or deeper neural networks for image SR [32, 53, 31, 5]. For example, RCAN [53] is a representative CNN-based image SR network with 15.59M parameters and reaching a depth of over 400 layers. One of the most significant drawbacks of these large models is that they require high computational costs, which makes them

*Corresponding author

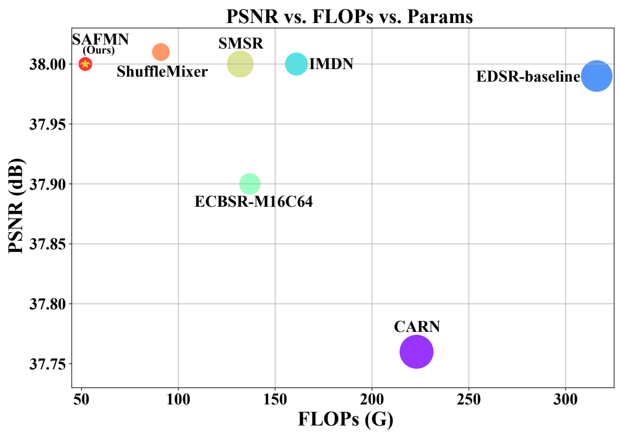


Figure 1. **Model complexity and performance comparison** between our proposed SAFMN model and other lightweight methods on Set5 [4] for ×2 SR. Circle sizes indicate the number of parameters. The proposed method achieves a better trade-off between model complexity and reconstruction performance.

challenging to deploy. Moreover, recent visual transformers (ViTs) [12, 31, 5] outperform convolutional neural networks (CNNs) in low-level vision tasks, and their results demonstrate that exploring non-local feature interactions is essential for high-quality reconstruction. But existing self-attention mechanisms are computationally expensive and unfriendly to efficient SR design. This, therefore, motivates us to develop a lightweight yet effective model for real-world applications of image super-resolution by integrating the principles of convolution and self-attention.

To reduce the heavy computational burden, various methods, including efficient module design [11, 41, 1, 20, 26, 33, 44, 45, 54, 30], knowledge distillation [16], neural architecture search [7], and structural re-parameterization [52], are trying to improve the efficiency of SR algorithms. Among these efficient SR models, one direction is to reduce model parameters or complexity (FLOPs). Lightweight strategies like recursive manner [23, 46], parameter sharing [1], and spare convolutions [1, 45, 30] are adopted. Although these approaches certainly reduce the model size, they usually compensate for the performance drop caused by shared recursive modules or sparse convolutions by increasing the depth or width of the model, which affects the inference efficiency when performing SR reconstruction.

Another direction is to accelerate the inference time. The post-upsampling [11, 44] is an important replacement for the pre-defined input [10, 24], which significantly speeds up the runtime. Model quantization [21] effectively accelerates latency and reduces energy consumption, particularly when deploying algorithms in edge-devices. Structural re-parameterization [52, 8] improves the speed of a well-trained model in the inference stage. These methods enjoy fast running time but poor reconstruction performance. Consequently, there is still room for a better trade-off between model efficiency and reconstruction performance.

To address the above-mentioned issues, we design a simple yet effective model by developing a spatially-adaptive feature modulation, namely SAFMN, to realize a favorable trade-off between performance and efficiency. Specifically, we first exploit non-local feature relations to dynamically select representative features (see Figure 2 and 5) by implementing a feature modulation mechanism based on multi-scale representations. As the modulation mechanism processes input features from a non-local perspective, there is a requirement to complement local contextual information. To this end, we present a convolutional channel mixer based on the FMBConv [47] to encode local features and mix channels. Taken together, we find that the SAFMN network is able to achieve a better trade-off between SR performance and model complexity, as shown in Figure 1.

The main contributions of this paper are summarized as follows:

- We develop an efficient feature modulation mechanism to learn feature dependencies, which absorbs CNN-like efficiency and transformer-like adaptability.
- We present a compact convolutional channel mixer that simultaneously encodes local contextual information and performs channel mixing.
- We evaluate the proposed method quantitatively and qualitatively on benchmark datasets, and the results show that our SAFMN achieves a favorable trade-off between accuracy and model complexity.

2. Related Work

Deep Learning-based Image Super-Resolution. Classical interpolation algorithms, such as linear or bicubic upsampling, create high-resolution images by inserting zeros between adjacent pixels in the low resolution and then using a low-pass filter to preserve the content information of the input image [41]. Unlike these interpolation-based upsamplers, deep learning-based approaches learn a nonlinear mapping between the input image and the target output in an end-to-end training fashion. SRCNN [10] is the first attempt to use a convolutional neural network to tackle the image SR problem and achieves a considerable performance gain compared with conventional methods. Since then, many improvements have been proposed. VDSR [24] uses global residual learning [15] to solve the problem of difficulty in training an SR model with deep layers. DRRN [46] integrates the local residual learning and global residual connection to ease the training difficulty and enhance high-frequency details. EDSR [32] further increases the model footprint to 43M, achieving a significant breakthrough in reconstruction performance and showing that the BatchNorm (BN) [22] layer is not necessary for the SR task. RCAN [53] builds a more than 400 layers model based on channel attention and dense connections for accurate SR. NLSA [40] utilizes a non-local sparse attention to enhance the feature modeling capability and reaches state-of-the-art performance. With the successful application of ViT in various high-level vision tasks, image SR also follows this ViT [12] framework, achieving higher reconstruction performance than CNN-based models on extensive public benchmarks. For instance, SwinIR [31] acts as a strong baseline for image restoration tasks based on the Swin Transformer [34]. While these approaches achieve impressive reconstruction performance, the required high computational costs make them challenging to deploy in real-world applications on resource-constrained devices.

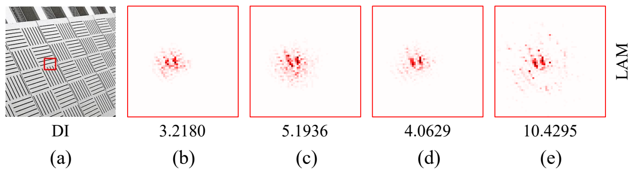


Figure 2. **Comparison of local attribution maps (LAMs) [13] and diffusion indices (DIs) [13]** between our SAFMN and other efficient SR models. (a) is the LR input, (b)-(e) are the results of CARN [1], EDSR-baseline [32], PAN [54] and SAFMN, respectively. The LAM results denote the importance of each pixel in the input LR image when super-resolving the patch marked with a red box. The DI value reflects the range of involved pixels. A larger DI value means a wider range of attention. The proposed method can exploit more feature information.

Efficient Image Super-Resolution. To improve the model efficiency, many CNN-based SR works try to alleviate this issue. FSRCNN [11] and ESPCN [44] utilize a post-upsampling manner to reduce the computational burden from the pre-defined inputs significantly. CARN [1] uses group convolutions and a cascading mechanism upon residual networks to improve efficiency. IMDN [20] adopts feature splitting and concatenation operations to progressively aggregate features, and its improved variants [33, 26] won the AIM2020 and NTIRE2022 Efficient SR challenge. ShuffleMixer [45] introduces a large kernel convolution for lightweight SR design. BSRN [30] employs blueprint-separable convolutions to reduce model complexity. Mean-

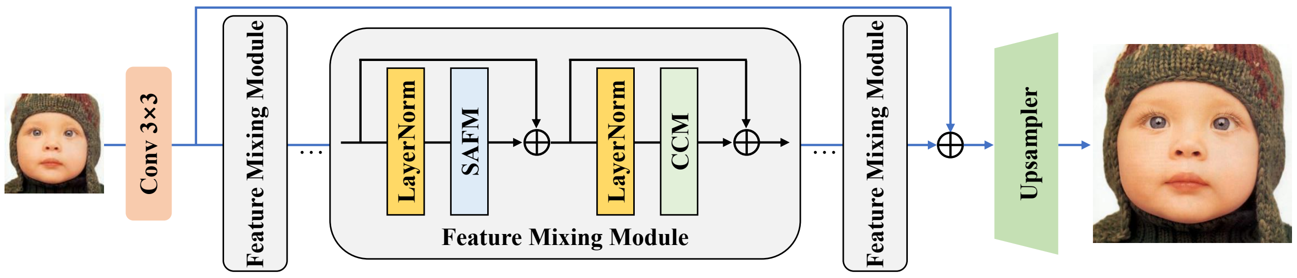


Figure 3. **An overview of the proposed SAFMN.** SAFMN first transforms the input LR image into the feature space using a convolutional layer, performs feature extraction using a series of feature mixing modules (FMMs), and then reconstructs these extracted features by an upsampler module. The FMM block is implemented by a spatially-adaptive feature modulation (SAFM) layer, a convolutional channel mixer (CCM) and two skip-connections.

while, an increasingly popular direction is to compress or accelerate a well-trained deep model through model quantization [21], structural re-parameterization [52] or knowledge distillation [16]. Neural architecture search (NAS) is also commonly used to search a well-constrained architecture for image super-resolution [7]. Note that the efficiency of a deep neural network could be measured in different metrics, including the number of parameters, FLOPs, activations, memory consumption and inference running time [29]. Although the above approaches have been improved in different efficiency aspects, there is still room for a favorable trade-off between reconstruction performance and model efficiency.

3. Proposed Method

In this section, we present the core components of our proposed model for efficient SISR. As shown in Figure 3, the network consists of the following parts: a shallow convolution, 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) layer and a convolutional channel mixer (CCM). 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 layer [44].

3.1. Spatially-adaptive Feature Modulation

Recent studies [9, 35, 14, 39, 31] suggest that the notable performance of ViTs across diverse tasks stems from their implementation of the key multi-head self-attention (MHSA) mechanism. This mechanism enables the model with the capability of long-range feature interaction and dynamic spatial weighting, both of which contribute to producing promising results. But existing self-attention variants are computationally expensive and unfriendly to efficient SR design. In contrast, standard convolution is an efficient operation, but its performance is limited by the static weights and locality. This observation inspires us to borrow the idea of MHSA to enhance the representation capability of convolution.

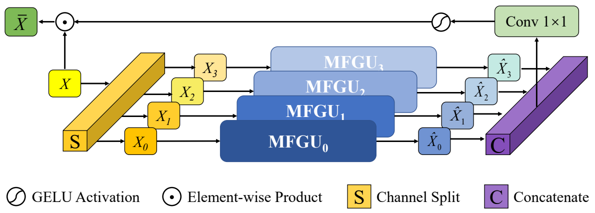


Figure 4. **Network architecture of the proposed SAFM module.** This module performs spatially-adaptive modulation on the input features. Please see the Section 3.1 for more details.

To introduce the ability of long-range interaction and dynamic modeling into convolution, we follow the multi-head paradigm, using parallel and independent computations that allow each head to process different scale information of the input, and then aggregate these features to generate an attention map for spatially-adaptive feature modulation.

The detailed architecture of our propoed SAFM is depicted in Figure 4. Specifically, we first split the normalized input features into four-group components, and feed them into a multi-scale feature generation unit (MFGU), where a 3×3 depth-wise convolution processes the first one, and the rest parts are sampled individually by pooling operations. As we desire to select discriminative features towards learning non-local feature interactions, adaptive max pooling operators are applied over the input features to gather information. Given the input feature X, this procedure can be formulated as:

$$\begin{aligned} &[X_0, X_1, X_2, X_3] = \text{Split}(X), \\ &\hat{X}_0 = \text{DW-Conv}_{3\times3}(X_0), \\ &\hat{X}_i = \uparrow_p (\text{DW-Conv}_{3\times3}(\downarrow_{\frac{p}{2^i}} (X_i))), 1 \leq i \leq 3, \end{aligned} \tag{1}$$

where Split$(\cdot)$ corresponds to the channel split operation, DW-Conv$_{3\times3}(\cdot)$ is a 3×3 depth-wise convolution, $\uparrow_p (\cdot)$ represents upsampling features at a specific level to the original resolution p via nearest interpolation for fast implementation, and $\downarrow_{\frac{p}{2^i}}$ denotes pooling the input features to the size of $\frac{p}{2^i}$. Figure 5(b)-(e) show that using such a design, we can obtain varying features from different representation subspaces.

Afterward, we aggregate these extracted short- or long-range features by concatenating them on the channel dimension and performing a 1×1 convolution. It can be formulated as:

$$\hat{X} = \text{Conv}_{1\times1}\left(\text{Concat}([\hat{X}_0, \hat{X}_1, \hat{X}_2, \hat{X}_3])\right), \quad (2)$$

where Concat$(\cdot)$ denotes the concatenation operation, and Conv$_{1\times1}(\cdot)$ is the 1×1 convolution. After obtaining the aggregated representation $\hat{X}$ (see Figure 5(f)), we normalize it through a GELU non-linearity [17] to estimate the attention map and adaptively modulate the input X according to the estimated attention via element-wise product. This process can be written as:

$$\bar{X} = \phi(\hat{X}) \odot X, \quad (3)$$

where $\phi(\cdot)$ represents the GELU function and $\odot$ is the element-wise product. The benefits of those operators are compared and results are shown in Table 3.

3.2. Feature Mixing Module

As mentioned previously, the SAFM module prioritizes exploiting non-local feature interactions. To further incorporate local contextual information and perform channel mixing simultaneously, we introduce a compact convolutional channel mixer (CCM) based on the FMBConv [47]. The CCM comprises a 3×3 convolution and a 1×1 convolution. Within this, the first 3×3 convolution encodes the spatially local contexts and **doubles** the number of channels of the input features for mixing channels; the later 1×1 convolution reduces the channels back to the original input dimension. A GELU [17] function is applied to the hidden layer for non-linear mapping.

We formulate the proposed SAFM and CCM into a unified feature mixing module (FMM) to select representative features. The FMM can be written as:

$$\begin{aligned} Y &= \text{SAFM}(\text{LN}(X)) + X, \\ Z &= \text{CCM}(\text{LN}(Y)) + Y, \end{aligned} \quad (4)$$

where LN$(\cdot)$ is the LayerNorm [3] layer, X, Y, and Z are the intermediate features. The additional residual learning is employed to stabilize training processes and learn high frequency details to facilitate high-quality image reconstruction.

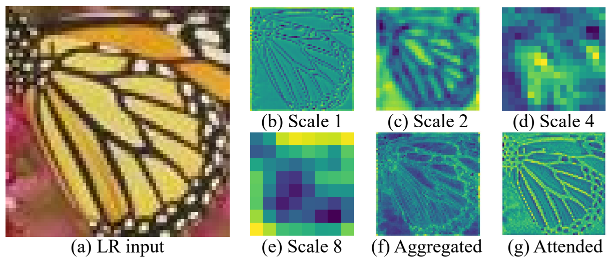


Figure 5. **Illustration of learned deep features from the SAFM module.** (a) is the LR input. (b)-(e) are features learned by the multi-scale feature generation unit. (f) is the results of aggregating features (b)-(e). (g) denotes the attended output of the SAFM.

4. Experimental Results

In this section, we perform quantitative and qualitative evaluations with state-of-the-art algorithms to demonstrate the effectiveness of the proposed method.

4.1. Dataset and Implementation

Datasets. Following previous works [28, 31, 45], we use DIV2K [48] and Flickr2K [32] as the training data and generate LR images by applying the bicubic downscaling to reference HR images. We use five commonly used benchmark datasets inluding Set5 [4], Set14 [51], B100 [2], Urban100 [19] and Manga109 [38] as test data. We use the peak signal to noise ratio (PSNR) and the structural similarity index (SSIM) to evaluate the quality of the restored images. All PSNR and SSIM values are calculated on the **Y** channel of images transformed to YCbCr color space.

Implementation details. During the training, the data argumentation is performed on the input patches with random horizontal flips and rotations. In addition, we randomly crop 64 patches of size 64×64 pixels from LR images as the basic training inputs. The number of FMM and feature channels is set to 8 and 36, respectively. We use the Adam [25] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to solve the proposed model. The number of iterations is set to 500,000. We set the initial learning rate to 1×10^{-3} and the minimum one to 1×10^{-5}, which is updated by the Cosine Annealing scheme [36]. We use a combination of mean absolute error (MAE) loss and an FFT-based frequency loss function to constrain the model training, which is the same as [45]. All experiments are conducted with the PyTorch framework on an NVIDIA GeForce RTX 3090 GPU. Due to the page limit, we include more comparison results in the supplemental material.

4.2. Comparisons with State-of-the-Art Methods

Quantitative comparisons. To evaluate the performance of the proposed approach, we compare it with state-of-the-art lightweight SR methods on different scaling factors, including SRCNN [10], FSRCNN [11], ESPCN [44], VDSR [24], LapSRN [27], CARN [1], EDSR-baseline [32], IMDN [20],

Table 1. **Comparisons of efficient SR networks on the commonly used benchmark datasets.** All PSNR/SSIM results are calculated on the **Y**-channel. #Acts represents all elements of the output of convolutional layers. #FLOPs and #Acts are measured corresponding to an HR image of the size 1280 × 720 pixels. Red color denotes the best performance. Blanked entries indicate results not reported or not available from previous work.

Methods	Scale	#Params [K]	#FLOPs [G]	#Acts [M]	Set5	Set14	B100	Urban100	Manga109
Bicubic	×2	-	-	-	33.66/0.9299	30.24/0.8688	29.56/0.8431	26.88/0.8403	30.80/0.9339
SRCNN [10]		57	53	89	36.66/0.9542	32.42/0.9063	31.36/0.8879	29.50/0.8946	35.74/0.9661
FSRCNN [11]		12	6	41	37.00/0.9558	32.63/0.9088	31.53/0.8920	29.88/0.9020	36.67/0.9694
ESPCN [44]		21	5	23	36.83/0.9564	32.40/0.9096	31.29/0.8917	29.48/0.8975	-
VDSR [24]		665	613	1,120	37.53/0.9587	33.03/0.9124	31.90/0.8960	30.76/0.9140	37.22/0.9729
LapSRN [27]		813	30	223	37.52/0.9590	33.08/0.9130	31.80/0.8950	30.41/0.9100	37.27/0.9740
CARN-M [1]		415	91	655	37.53/0.9583	33.26/0.9141	31.92/0.8960	31.23/0.9193	-
CARN [1]		1,592	223	522	37.76/0.9590	33.52/0.9166	32.09/0.8978	31.92/0.9256	-
EDSR-baseline [32]		1,370	316	563	37.99/0.9604	33.57/0.9175	32.16/0.8994	31.98/0.9272	38.54/0.9769
IMDN [20]		694	159	423	38.00/0.9605	33.63/0.9177	32.19/0.8996	32.17/0.9283	38.88/0.9774
PAN [54]		261	71	677	38.00/0.9605	33.59/0.9181	32.18/0.8997	32.01/0.9273	38.70/0.9773
LAPAR-A [28]		548	171	656	38.01/0.9605	33.62/0.9183	32.19/0.8999	32.10/0.9283	38.67/0.9772
ECBSR-M16C64 [52]		596	137	252	37.90/0.9615	33.34/0.9178	32.10/0.9018	31.71/0.9250	-
SMSR [49]		985	132	-	38.00/0.9601	33.64/0.9179	32.17/0.8990	32.19/0.9284	38.76/0.9771
ShuffleMixer [45]		394	91	832	38.01/0.9606	33.63/0.9180	32.17/0.8995	31.89/0.9257	38.83/0.9774
SAFMN (Ours)		**228**	**52**	**299**	38.00/0.9605	33.54/0.9177	32.16/0.8995	31.84/0.9256	38.71/0.9771
Bicubic	×3	-	-	-	30.39/0.8682	27.55/0.7742	27.21/0.7385	24.46/0.7349	26.95/0.8556
SRCNN [10]		57	53	89	32.75/0.9090	29.28/0.8209	28.41/0.7863	26.24/0.7989	30.59/0.9107
FSRCNN [11]		12	5	19	33.16/0.9140	29.43/0.8242	28.53/0.7910	26.43/0.8080	30.98/0.9212
VDSR [24]		665	613	1,120	33.66/0.9213	29.77/0.8314	28.82/0.7976	27.14/0.8279	32.01/0.9310
CARN-M [1]		415	46	327	33.99/0.9236	30.08/0.8367	28.91/0.8000	27.55/0.8385	-
CARN [1]		1,592	119	268	34.29/0.9255	30.29/0.8407	29.06/0.8034	28.06/0.8493	-
EDSR-baseline [32]		1,555	160	285	34.37/0.9270	30.28/0.8417	29.09/0.8052	28.15/0.8527	33.45/0.9439
IMDN [20]		703	72	190	34.36/0.9270	30.32/0.8417	29.09/0.8046	28.17/0.8519	33.61/0.9445
PAN [54]		261	39	340	34.40/0.9271	30.36/0.8423	29.11/0.8050	28.11/0.8511	33.61/0.9448
LAPAR-A [28]		594	114	505	34.36/0.9267	30.34/0.8421	29.11/0.8054	28.15/0.8523	33.51/0.9441
SMSR [49]		993	68	-	34.40/0.9270	30.33/0.8412	29.10/0.8050	28.25/0.8536	33.68/0.9445
ShuffleMixer [45]		415	43	404	34.40/0.9272	30.37/0.8423	29.12/0.8051	28.08/0.8498	33.69/0.9448
SAFMN (Ours)		**233**	**23**	**134**	34.34/0.9267	30.33/0.8418	29.08/0.8048	27.95/0.8474	33.52/0.9437
Bicubic	×4	-	-	-	28.42/0.8104	26.00/0.7027	25.96/0.6675	23.14/0.6577	24.89/0.7866
SRCNN [10]		57	53	89	30.48/0.8628	27.49/0.7503	26.90/0.7101	24.52/0.7221	27.66/0.8505
FSRCNN [11]		12	5	11	30.71/0.8657	27.59/0.7535	26.98/0.7150	24.62/0.7280	27.90/0.8517
ESPCN [44]		25	1	6	30.52/0.8697	27.42/0.7606	26.87/0.7216	24.39/0.7241	-
VDSR [24]		665	613	1,120	31.35/0.8838	28.01/0.7674	27.29/0.7251	25.18/0.7524	28.83/0.8809
LapSRN [27]		813	149	264	31.54/0.8850	28.19/0.7720	27.32/0.7280	25.21/0.7560	29.09/0.8845
CARN-M [1]		415	33	227	31.92/0.8903	28.42/0.7762	27.44/0.7304	25.62/0.7694	-
CARN [1]		1,592	91	194	32.13/0.8937	28.60/0.7806	27.58/0.7349	26.07/0.7837	-
EDSR-baseline [32]		1,518	114	202	32.09/0.8938	28.58/0.7813	27.57/0.7357	26.04/0.7849	30.35/0.9067
IMDN [20]		715	41	108	32.21/0.8948	28.58/0.7811	27.56/0.7353	26.04/0.7838	30.45/0.9075
PAN [54]		272	28	238	32.13/0.8948	28.61/0.7822	27.59/0.7363	26.11/0.7854	30.51/0.9095
LAPAR-A [28]		659	94	452	32.15/0.8944	28.61/0.7818	27.61/0.7366	26.14/0.7871	30.42/0.9074
ECBSR-M16C64 [52]		603	35	64	31.92/0.8946	28.34/0.7817	27.48/0.7393	25.81/0.7773	-
SMSR [49]		1006	42	-	32.12/0.8932	28.55/0.7808	27.55/0.7351	26.11/0.7868	30.54/0.9085
ShuffleMixer [45]		411	28	269	32.21/0.8953	28.66/0.7827	27.61/0.7366	26.08/0.7835	30.65/0.9093
SAFMN (Ours)		**240**	**14**	**77**	32.18/0.8948	28.60/0.7813	27.58/0.7359	25.97/0.7809	30.43/0.9063

PAN [54], LAPAR [28], ECBSR [52], SMSR [49], and ShuffleMixer [45].

The quantitative comparisons on benchmark datasets for the upscaling factors of ×2, ×3, and ×4 are reported in Table 1. In addition to PSNR/SSIM metrics, we list the number of parameters (#Params), FLOPs (#FLOPs) and activations (#Acts). We calculate the number of FLOPs and activations with the fvcore[1] library under a setting of super-resolving an LR image to 1280 × 720 pixels. Among these metrics, #Params and #Acts are linked to memory consumption, and #FLOPs is related to energy usage. In particular, #Acts is a better metric for measuring the efficiency of a model than the number of parameters and FLOPs as suggested in recent works [42, 52, 29].

Benefiting from the simple yet efficient structure, the proposed SAFMN obtains comparable performance with significantly fewer parameters and memory consumption. For instance, the ×4 SR results on the B100 dataset reveal that our SAFMN has parameters about 85% less than the CARN [1], 66% less than the IMDN [20], and 42% less than the ShuffleMixer [45]. As for activations, we have 60%, 29% and 71% fewer than them, respectively. While our model has a smaller footprint, we achieve similar performance among these methods. Moreover, we compare

[1]We use the fvcore.nn.flop_count_str command to calculate the number of parameters, FLOPs and activations.

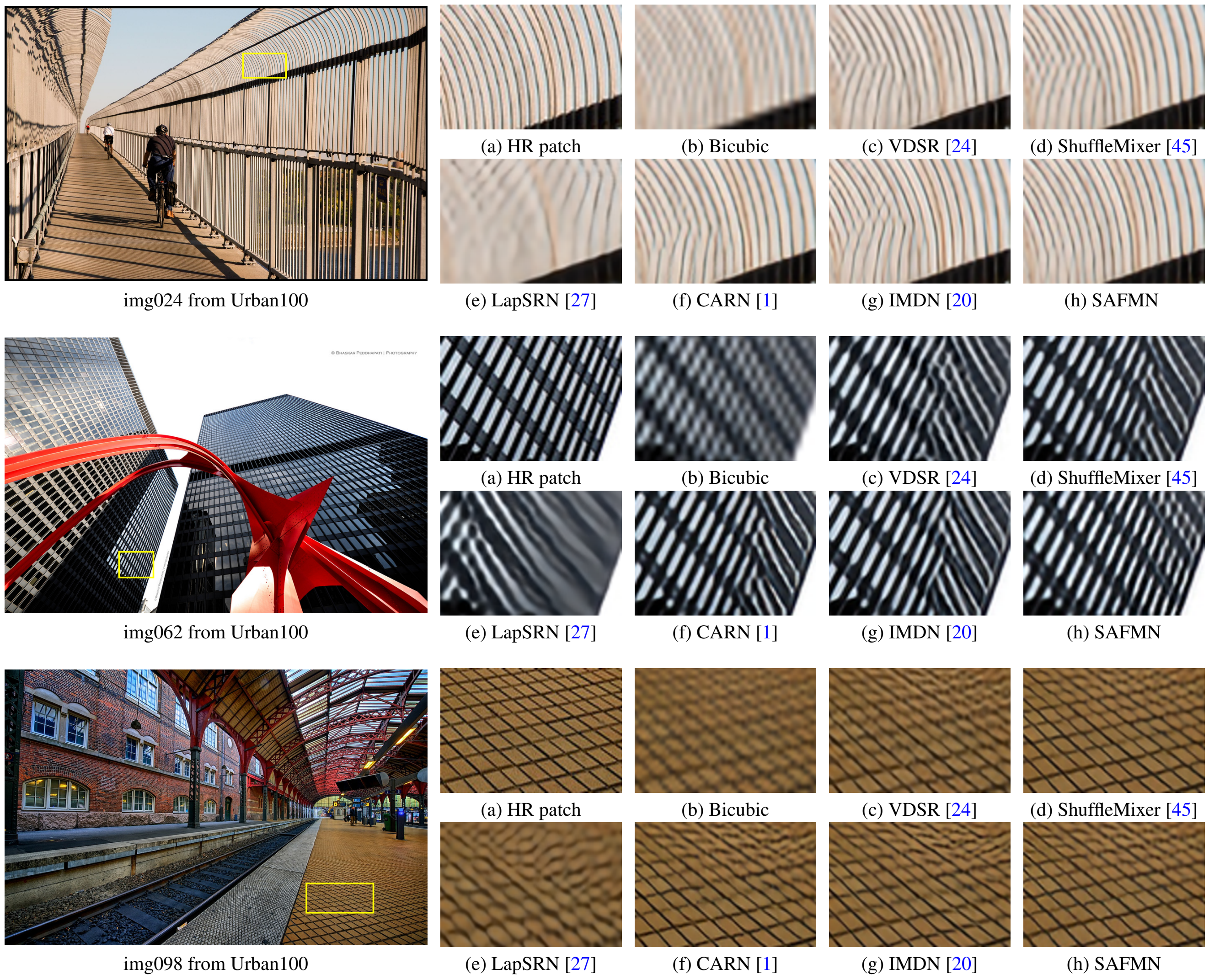


Figure 6. **Visual comparisons for $\times3$ SR on the Urban100 dataset.** The proposed method recovers the image with clearer structures.

the reconstruction accuracy, FLOPs and parameters on the $\times2$ Set5 dataset in Figure 1. The proposed SAFMN model achieves a favorable trade-off between model complexity and reconstruction performance.

Qualitative comparisons. In addition to the quantitative evaluations, we provide qualitative comparisons of the proposed method. Figure 6 shows the visual comparisons on the Urban100 dataset for $\times3$ SR. Our approach generates parallel straight lines and grid patterns more accurately than the listed methods. These results also demonstrate the effectiveness of our method for adaptive feature modulation by exploiting non-local feature interactions.

Memory and running time comparisons. To fully examine the efficiency of our proposed method, we further evaluate our method against five representative ones, including CARN-M [1], CARN [1], EDSR-baseline [32], IMDN [20], and LAPAR-A [28] on $\times4$ SR in terms of the GPU memory consumption (#GPU Mem.) and running time (#Avg. Time). The maximum GPU memory consumption is recorded during the inference. The running time is averaged on 50 test images with a resolution of 320×180 pixels. We show the memory and running time comparisons in Table 2, our method achieves a clear improvement over other state-of-the-art methods. By using the spatially-adaptive feature modulation layer and the compact channel mixer, the GPU consumption of our SAFMN is only 10% of the CARN series and 4% of LAPAR-A; the running time is nearly twice as fast as other evaluated methods, except for IMDN. Compared to IMDN [20], our method has a similar running time speed while significantly reducing memory usage. Tables 1 and 2 show that the proposed model achieves a favorable trade-off in terms of inference speed, model complexity and reconstruction performance against state-of-the-art methods.

Table 2. **Memory and running time comparisons on $\times 4$ SR.** #GPU Mem. denotes the maximum GPU memory consumption during the inference phase, derived with the Pytorch torch.cuda.max_memory_allocated() function. #Avg. Time is the average running time on 50 LR images of 320×180 pixels.

Methods	#GPU Mem. [M]	#Avg.Time [ms]
CARN-M [1]	680.84	17.85
CARN [1]	689.83	18.90
EDSR-baseline [32]	486.58	19.81
IMDN [20]	203.44	10.22
LAPAR-A [28]	1811.47	24.91
SAFMN	65.26	10.71

5. Analysis and Discussion

We further conduct extensive ablation studies to better understand and evaluate each component in the proposed SAFMN. For fair comparisons with the designed baselines, we implement all experiments based on $\times 4$ SAFMN and train them with the same setting. The experimental results in Table 3 are measured on DIV2K-val [48] and Manga109 [38] datasets.

Effectiveness of the SAFM. To demonstrate the effect of the spatially-adaptive feature modulation, we first remove this module for comparison. Without it, the PSNR values will drop by 0.17dB (30.26 vs. 30.43) and 0.34dB (30.09 vs. 30.43) on the DIV2K-val and Manga109 datasets, respectively. These results show the importance of the SAFM. Therefore, we further investigate this module to figure out why it works.

- **Feature modulation.** The feature modulation mechanism is introduced to enable the network with adaptive properties. Without this operation, the baseline model is lowered by 0.11dB on the Manga109 dataset.
- **Multi-scale representation.** Here, "w/o MR" in Table 3 denotes that we directly use a depth-wise convolution with a kernel size of 3×3 pixels to extract spatial information and do not employ multi-scale features. A noticeable performance drop of 0.13dB on the Manga109 dataset is observed without these multi-scale features. Moreover, we apply the adaptive max pooling over the input features in this module to build feature pyramids. Compared to using adaptive average pooling or nearest interpolation, adaptive max pooling allows the model to detect discriminative features, resulting in better reconstruction results.
- **Feature aggregation.** We use a 1×1 convolution to aggregate the multi-scale features on the channel dimension. Combined with the modulation mechanism, it brings a PSNR improvement of 0.14dB on the Manga109 dataset, which proves the necessity of aggregating the multi-scale features.

  Without all three components mentioned before, it represents that only a 3×3 depth-wise convolution is used to encode the spatial information and leads to a PSNR reduction of 0.12dB and 0.2dB on the DIV2K-val and Manga109 datasets, respectively. This performance drop suggests that the spatially-adaptive modulation based on the multi-scale feature representation effectively boosts SR reconstruction performance.
- **GELU function.** Here, we use the GELU [17] function to normalize the modulation map. Table 3 shows that better experimental results can be achieved with GELU than with Sigmoid or without GELU, mainly because it weights the input features by their percentile and allows better-activated features.

In addition, we use the local attribution map (LAM) [13] and diffusion index (DI) [13] to illustrate the feature interaction range. Figure 2 clearly shows that the proposed model can utilise more feature information than previous CNN-based efficient SR methods. The more information used, the better the reconstruction performance in general.

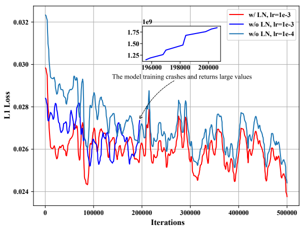
The above analysis shows that benefiting from the multi-scale feature representation, our proposed SAFM can effectively exploit non-local feature interactions with small memory and computational costs.

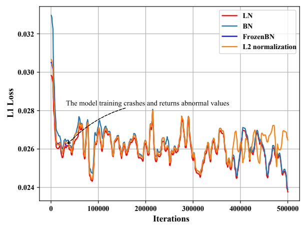
Effectiveness of the CCM. Compared with the original FMBConv [47], the main change made by CCM is removing the SE [18] block. As shown in Table 3, using SE blocks, its performance is almost the same as without them. This result is mainly caused by the SAFM block allows adaptability in channel and spatial dimensions. Hence, we do not use the SE blocks for saving parameters. We next conduct a series of ablations to verify that CCM can effectively encode local contextual information and perform channel mixing. Without it, the model only achieves the accuracy of 29.69dB and 28.49dB on DIV2K-val and Manga109 datasets, proving the indispensability and locality modelling ability of this part. We then change CCM to other channel mixers commonly used in ViT architectures: channel MLP [12] or inverted residual block [43]. When the channel MLP is adopted for the channel mixer, there is a significant performance drop of 0.63dB (29.80 vs. 30.43) on the Manga109 dataset. This performance reduction results from a lack of capacity for modeling local contextual information. For the inverted residual block, the performance is highly similar to that of CCM, but the corresponding #Acts is increased by nearly 33M, which means more memory consumption and slower inference time. Thus, we use the CCM as the default channel mixer.

Effect of the LayerNorm layer. Since we use the element-wise product in the SAFM module, it will result in abnormal gradient values and unstable model training, as shown in Figure 7(a). It is, therefore, necessary to normalize the input features. Here, we perform normalization with the LayerNorm [3] layer. To verify this assumption, we first remove the LayerNorm layer. Table 3 and Figure 7(a) show

Table 3. **Ablation for SAFMN on DIV2K-val and Manga109 datasets.** SAFMN with a scaling factor of $\times 4$ is utilized as the baseline for ablation studies. The PSNR/SSIM values on benchmarks are reported. "A $\rightarrow$ B" is to replace A with B. "None" means to remove the operation. "lr" denotes the learning rate. "FBN" is the abbreviation of the Frozen BatchNorm. "L_2 normalization" represents that the inputs are normalized by L_2-norm over the channel dimension. "FM", "MR", and "FA" are the abbreviations corresponding to feature modulation, multi-scale representation, and feature aggregation. * indicates that the corresponding results are obtained before the training collapse. The number of parameters, FLOPs and Acts are calculated in the same way as in Table 1.

Ablation	Variant	#Params [K]	#FLOPs [G]	#Acts [M]	DIV2K_val	Manga109
Baseline	-	**239.52**	**13.56**	**76.70**	**30.43/0.8372**	**30.43/0.9063**
Main module	SAFM $\rightarrow$ None	225.41	12.90	54.61	30.26/0.8330	30.09/0.9018
	CCM $\rightarrow$ None	30.72	1.61	26.93	29.69/0.8193	28.49/0.8756
SAFM	(a): w/o FM	239.52	13.56	76.70	30.36/0.8357	30.32/0.9048
	(b): w/o MR	239.52	13.64	87.78	30.34/0.8350	30.30/0.9047
	(c): w/o FA	228.86	12.96	60.11	30.36/0.8355	30.29/0.9049
	(a) + (b)	239.52	13.64	87.78	30.32/0.8345	30.24/0.9038
	(a) + (c)	228.86	12.96	60.11	30.34/0.8351	30.28/0.9043
	(a) + (b) + (c)	228.86	13.05	71.19	30.31/0.8344	30.23/0.9036
	AdaptiveMaxPool $\rightarrow$ AdaptiveAvgPool	239.52	13.56	76.70	30.40/0.8364	30.41/0.9061
	AdaptiveMaxPool $\rightarrow$ Nearest interpolate	239.52	13.56	76.70	30.36/0.8354	30.31/0.9048
	GELU $\rightarrow$ None	239.52	13.56	76.70	30.40/0.8366	30.37/0.9058
	GELU $\rightarrow$ Sigmoid	239.52	13.56	76.70	30.36/0.8355	30.29/0.9044
CCM	w/ SE	260.98	13.59	76.70	30.39/0.8360	30.46/0.9067
	CCM $\rightarrow$ Channel MLP	73.63	4.00	76.70	30.17/0.8313	29.80/0.8980
	CCM $\rightarrow$ Inverted residual block	245.28	13.85	110.00	30.43/0.8373	30.43/0.9064
Normalization	LN $\rightarrow$ None, lr=$1 \times 10^{-3*}$	238.37	13.55	76.70	30.29/0.8340	30.04/0.9014
	LN $\rightarrow$ None, lr=1×10^{-4}	238.37	13.55	76.70	30.15/0.8306	29.74/0.8970
	LN $\rightarrow$ BN	239.52	13.72	76.70	30.28/0.8354	30.05/0.9029
	LN $\rightarrow$ FBN*	238.37	13.55	76.70	30.30/0.8343	30.15/0.9028
	LN $\rightarrow$ L_2 normalization	238.37	13.55	76.70	30.37/0.8358	30.31/0.9049





(a) Effect of LayerNorm (b) Comparison of different normalizations

Figure 7. **Smoothed training loss curves with or without normalization.** Figure (a) shows LayerNorm stabilizes model training and converges better than without it. Figure (b) compares the effects of different normalization approaches on model training.

that without it, the model suffers a training crash at a large learning rate and does not converge well at small ones with its PSNR of only 30.15dB and 29.74dB on the DIV2K-val and Manga109 datasets, respectively. Next, we compare the LayerNorm with three representative normalization methods, including BatchNorm [22], Frozen BatchNorm (FBN) [22], and L_2 normalization in Figure 7(b). The results in Table 3 and Figure 7(b) demonstrate that the BN family decrease the PSNR/SSIM values by a large margin, and the FBN even does not guarantee a stable model training process. Although the L_2 normalization allows the model to be trained successfully, its performance is not as good as that of using LayerNorm. Thus, the LayerNorm layer is set as default for the proposed SAFMN.

Table 4. **Effect of the number of FMMs on performance.**

#N	#Params [K]	#FLOPs [G]	#Acts [M]	DIV2K_val	Manga109
4	128.06	7.25	40.77	30.21/0.8319	29.86/0.8988
6	183.79	10.40	58.64	30.32/0.8347	30.18/0.9033
8	239.52	13.56	76.70	30.43/0.8372	30.43/0.9063
12	350.97	19.86	113.00	30.51/0.8389	30.71/0.9096

Table 5. **GPU runtime of meta-operations in the SAFM layer.**

Meta-operations	LayerNorm	Channel Splitting	Adaptive MaxPool	Nearest Interpolation	Element-wise Product	Element-wise Addition
Runtime [us]	465	133	85	24	156	193

Effect of the number of FMMs. The proposed network contains several FMMs. We evaluate the effect of the number of FMMs by setting the number #N from 4 to 12. Table 4 shows that the performance gradually improves as #N increases. To achieve a favorable trade-off between model complexity and reconstruction performance, we choose 8 blocks as the default setting.

Relations to lightweight ViT models. To alleviate the quadratic complexity of vanilla ViT [12], MobileViT [39], Mobile-Former [6], and EdgeNeXt [37] adopt a hybrid CNN-Transformer design, which replaces the shallow transformer blocks with convolutional layers to avoid using self-attention on high-resolution representations, and efficiently combines the transformer blocks with the modified mobilenet blocks to perform feature interactions in the later stages. Another alternative [50, 14] is to incorporate concepts from the transformer into convolutional modules. For example, MetaFormer [50] and VAN [14] use pooling operation or large kernel attention to replace the heavy-weight self-attention for contextual information aggregation. Compared to these approaches, we develop an efficient feature modulation mechanism upon a grouped multi-scale representation to learn feature dependencies. Benefiting from the multi-head paradigm, the proposed model can exploit multi-scale contexts across different heads for adaptive feature modulation while maintaining a low computational cost.

Limitations. Although our SAFMN has better overall performance than the evaluated methods [1, 20, 49, 28, 45], its running time is relatively slow. As shown in Table 5, we count the inference time of different commonly used tensor operations with the torch.profiler function, and the results show that the utilized LayerNorm layers and residual connections lead to increased runtime. The running time of SAFMN can be further accelerated by optimizing the implementation of these components. We will investigate these in the future work. Another limitation is that the PSNR performance of SAFMN is suboptimal, especially on the Urban100 dataset, which is mainly due to the fact that we perform the feature modulation without considering the integration of local features.

6. Conclusion

In this paper, we propose a simple yet efficient CNN model to solve the efficient image super-resolution problem. Our SAFMN exploits non-local feature interactions upon a multi-scale feature representation. To complement the local contextual information, we further develop a compact convolutional channel mixer to encode spatially local context and conduct channel mixing simultaneously. We both qualitatively and quantitatively evaluate the proposed method on commonly used benchmarks. Experimental results show that the proposed SAFMN model is more efficient than state-of-the-art methods while achieving competitive performance.

Acknowledgements. This work has been partly supported by the National Key R&D Program of China (No. 2018AAA0102001), the National Natural Science Foundation of China (Nos. U22B2049, 62272233, 61922043, 61925204), and the Fundamental Research Funds for the Central Universities (Nos. 30922010910, 30920041109).

References

[1] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, 2018. 1, 2, 4, 5, 6, 7, 9

[2] Pablo Arbeláez, Michael Maire, Charless C. Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011. 4

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4, 7

[4] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie line Alberi Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *BMVC*, 2012. 1, 4

[5] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *CVPR*, 2021. 1

[6] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *CVPR*, 2022. 8

[7] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. In *ICPR*, 2021. 1, 3

[8] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *CVPR*, 2021. 2

[9] 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 *CVPR*, 2022. 3

[10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *PAMI*, 38(2):295–307, 2016. 2, 4, 5

[11] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016. 1, 2, 4, 5

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2, 7, 8

[13] Jinjin Gu and Chao Dong. Interpreting super-resolution networks with local attribution maps. In *CVPR*, 2021. 2, 7

[14] 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. 3, 8

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[16] Zibin He, Tao Dai, Jian Lu, Yong Jiang, and Shu-Tao Xia. Fakd: Feature-affinity based knowledge distillation for efficient image super-resolution. In *ICIP*, 2020. 1, 3

[17] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units. *arXiv preprint arXiv:1606.08415*, 2016. 4, 7

[18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 7

[19] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, 2015. 4

[20] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACM MM*, 2019. 1, 2, 4, 5, 6, 7, 9

[21] Andrey Ignatov, Radu Timofte, Maurizio Denna, and et al. Efficient and accurate quantized image super-resolution on Mobile NPUs, Mobile AI & AIM 2022 challenge: Report. In *ECCV Workshops*, 2022. 2, 3

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2, 8

[23] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016. 1

[24] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 2, 4, 5, 6

[25] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[26] 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 *CVPR Workshops*, 2022. 1, 2

[27] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017. 4, 5, 6

[28] Wenbo Li, Kun Zhou, Lu Qi, Nianjuan Jiang, Jiangbo Lu, and Jiaya Jia. LAPAR: Linearly-assembled pixel-adaptive regression network for single image super-resolution and beyond. In *NeurIPS*, 2020. 4, 5, 6, 7, 9

[29] Yawei Li, Kai Zhang, Luc Van Gool, Radu Timofte, et al. NTIRE 2022 challenge on efficient super-resolution: Methods and results. In *CVPR Workshops*, 2022. 3, 5

[30] 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 *CVPR Workshops*, 2022. 1, 2

[31] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. SwinIR: Image restoration using swin transformer. In *ICCV Workshops*, 2021. 1, 2, 3, 4

[32] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshops*, 2017. 1, 2, 4, 5, 6, 7

[33] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *ECCV Workshops*, 2020. 1, 2

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 2

[35] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *CVPR*, 2022. 3

[36] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 4

[37] Muhammad Maaz, Abdelrahman Shaker, Hisham Cholakkal, Salman Khan, Syed Waqas Zamir, Rao Muhammad Anwer, and Fahad Shahbaz Khan. Edgenext: Efficiently amalgamated cnn-transformer architecture for mobile vision applications. In *ECCV Workshops*, 2022. 8

[38] Yusuke Matsui, Kota Ito, Yuji Aramaki, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *arXiv preprint arXiv:1510.04389*, 2015. 4, 7

[39] Sachin Mehta and Mohammad Rastegari. MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer. In *ICLR*, 2022. 3, 8

[40] Yiqun Mei, Yuchen Fan, and Yuqian Zhou. Image super-resolution with non-local sparse attention. In *CVPR*, 2021. 2

[41] Pablo Navarrete Michelini, Yunhua Lu, and Xingqun Jiang. edge–SR: Super–resolution for the masses. In *WACV*, 2022. 1, 2

[42] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. In *CVPR*, 2020. 5

[43] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 7

[44] 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. 1, 2, 3, 4, 5

[45] Long Sun, Jinshan Pan, and Jinhui Tang. ShuffleMixer: An efficient convnet for image super-resolution. In *NeurIPS*, 2022. 1, 2, 4, 5, 6, 9

[46] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. *CVPR*, 2017. 1, 2

[47] Mingxing Tan and Quoc Le. EfficientNetV2: Smaller models and faster training. In *ICML*, 2021. 2, 4, 7

[48] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, Lei Zhang, et al. NTIRE 2017 challenge on single image super-resolution: Methods and results. In *CVPR Workshops*, 2017. 4, 7

[49] Longguang Wang, Xiaoyu Dong, Yingqian Wang, Xinyi Ying, Zaiping Lin, Wei An, and Yulan Guo. Exploring sparsity in image super-resolution for efficient inference. In *CVPR*, 2021. 5, 9

[50] 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 *CVPR*, 2022. 8

[51] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *Curves and Surfaces*, 2012. 4

[52] Xindong Zhang, Hui Zeng, and Lei Zhang. Edge-oriented convolution block for real-time super resolution on mobile devices. In *ACM MM*, 2021. 1, 2, 3, 5

[53] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018. 1, 2

[54] Hengyuan Zhao, Xiangtao Kong, Jingwen He, Yu Qiao, and Chao Dong. Efficient image super-resolution using pixel attention. In *ECCV Workshops*, 2020. 1, 2, 5