Fast and Memory-Efficient Network Towards Efficient Image Super-Resolution

Zongcai Du[1], Ding Liu[2], Jie Liu[1], Jie Tang[1], Gangshan Wu[1], Lean Fu[2]
[1]State Key Laboratory for Novel Software Technology, Nanjing University, China
[2]ByteDance Inc.
{151220022, jieliu}@smail.nju.edu.cn, {tangjie, gswu}@nju.edu.cn,
liudingdavy@gmail.com, fulean@bytedance.com

Abstract

Runtime and memory consumption are two important aspects for efficient image super-resolution (EISR) models to be deployed on resource-constrained devices. Recent advances in EISR [16, 32] exploit distillation and aggregation strategies with plenty of channel split and concatenation operations to fully use limited hierarchical features. In contrast, sequential network operations avoid frequently accessing preceding states and extra nodes, and thus are beneficial to reducing the memory consumption and runtime overhead. Following this idea, we design our lightweight network backbone by mainly stacking multiple highly optimized convolution and activation layers and decreasing the usage of feature fusion. We propose a novel sequential attention branch, where every pixel is assigned an important factor according to local and global contexts, to enhance high-frequency details. In addition, we tailor the residual block for EISR and propose an enhanced residual block (ERB) to further accelerate the network inference. Finally, combining all the above techniques, we construct a fast and memory-efficient network (FMEN) and its small version FMEN-S, which runs 33% faster and reduces 74% memory consumption compared with the state-of-the-art EISR model: E-RFDN, the champion in [49]. Besides, FMEN-S achieves the lowest memory consumption and the second shortest runtime in NTIRE 2022 challenge on efficient super-resolution [28]. Code is available at https://github.com/NJU-Jet/FMEN.

1. Introduction

Single image super-resolution (SR) is a typical low-level vision problem, with the purpose of recovering a high-resolution (HR) image according to its degraded low-resolution (LR) counterpart. To solve this highly ill-posed problem, different kinds of methods have been proposed. Among them, deep learning based methods [9, 16, 31, 32, 51], represented by convolution neural network (CNN), have produced superior results and revolutionized this area.

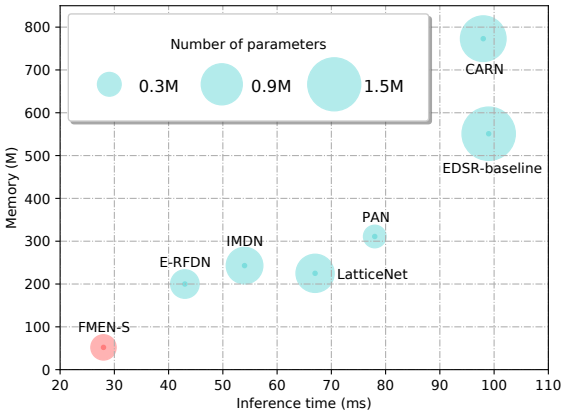


Figure 1. Comparison with recent EISR models. We show three metrics: inference time on an NVIDIA 1080Ti GPU, memory consumption of input of size 256×256 and the number of parameters for $\times 4$ upscaling, when all these models obtain comparable performance on DIV2K validation set (maintain the PSNR of 29.00dB).

Due to the realistic demand of resource-limited devices, efficient image super-resolution (EISR) attracts growing attention of the SR community. In the early stage, recurrent neural network [29, 40] and group convolution [1] are adopted to reduce the model parameters, but they bring about either high computation cost or inferior performance. Recently, many researchers concentrate on elaborate network design, since it not only exerts a direct influence on the performance but also affects the follow-up network pruning [3, 5, 26, 27] and knowledge distillation [36, 43]. The key point of efficient network design is how to sufficiently utilize limited features to generate more representative features.

We observe that recent EISR methods tend to apply feature fusion to achieve this goal [1, 16, 17, 32, 47] due to the following two main advantages. First, multi-level con-

nections are conductive to propagating gradients, which facilitates the training of deeper networks. Second, it is often associated with the gating mechanism which adaptively controls the previous and current states in feature domain. CARN [1] used cascading mechanism at both local and global levels to incorporate the features from multiple layers. IMDN [16] proposed information multi-distillation blocks (IMDB) to extract hierarchical features step-by-step and aggregate them according to the importance of candidate features. LatticeNet [47] adopted a backward sequential concatenation strategy for feature fusion of different receptive fields. RFDN [32] applied residual feature distillation block which is a variant of IMDB but more powerful and flexible. The feature fusion strategies mentioned above suffer from huge memory consumption that stem from multiple relevant feature maps residing in the memory until aggregated. In addition, the fusion design usually reduces inference speed because of introducing extra nodes (*e.g.*, concatenation, 1×1 convolution) and frequent memory access as discussed in RepVGG [8]. To accelerate inference speed and reduce memory consumption, we design our network backbone via highly optimized serial network operations instead of feature aggregation, and boost feature representations via attention mechanism.

Considering the goal of SR is to recover the lost high-frequency details (*e.g.*, edges, textures), we propose a high-frequency attention block (HFAB) which learns an attention map with special focus on the high-frequency area. Specifically, we design the attention branch in HFAB from local and global perspectives. We stack highly efficient operators like 3×3 convolution and Leaky ReLU layers sequentially for modeling the relationship between local signals. Batch Normalization (BN) is injected into HFAB to capture global context during training, while merged into convolution during inference. Furthermore, we tailor the residual block (RB) and introduce an enhanced residual block (ERB), where the features are extracted in higher dimensional space during training and the skip connections are removed during inference using structural re-parameterization technique [8]. We show that such a design is able to accelerate the network inference and reduce memory consumption without sacrificing SR performance, when comparing with conventional RB.

By applying ERB and HFAB in a sequential and alternative way, we construct an efficient network, namely fast and memory-efficient network (FMEN), which demonstrates the clear advantage over existing EISR methods in terms of runtime and peak memory consumption when maintaining the same level of restoration performance. Besides, we build a smaller model by reducing the number of convolution kernels, FMEN-S, which runs 33% faster and reduces 74% memory consumption compared with E-RFDN as shown in Fig. 1, our contributions are as follows:

- We carefully analyze the factors which influence the inference speed and memory consumption of EISR models.
- We propose a high-frequency attention block (HFAB) to enchance high-frequency features and an enhanced residual block (ERB) to utilize residual learning with faster inference and less memory consumption.
- We construct a fast and memory-efficient network by sequentially combining HFAB and ERB, which achieves the lowest memory consumption and the second shortest runtime in NTIRE 2022 challenge on efficient super-resolution.

2. Related Work

2.1. Overview of Image Super-Resolution

Recently, convolution neural network based (CNN-based) methods have attained excellent results in many tasks including SISR. Since Dong *et al*. [9] creatively introduced a three-layer end-to-end CNN called SRCNN to restore HR image, diverse methods have been proposed to further enhance learning capability. To reduce its high computational cost which is caused by learning in HR space, shi *et al*. designed ESPCN [38] to replace the bicubic filter with sub-pixel convolution, which is adopted by mainstream SR architectures [4, 32, 44, 47, 50]. In the same period, Kim *et al*. [19] used residual learning and adjustable gradient clipping to train an extremely accurate twenty-layer model, demonstrating that depth plays an important role in the reconstruction performance. Subsequently, Ledig *et al*. [24] successfully exploited residual block (RB) [12] and presented a GAN-based network to recover photo-realistic textures. Furthermore, based on SRResNet [24], Lim *et al*. [31] proposed an enhanced deep super-resolution network (EDSR), which surpassed previous networks by removing unnecessary modules in RB and inspired succeeding works [1, 33, 47, 51–53]. For instance, RDN [53] presented residual dense block to make full use of all the hierarchical features via dense connected convolution layers. RCAN [51] integrated channel attention mechanism into RB and adopted residual-in-residual (RIR) structure to form a very deep network. LatticeNet [47] applied butterfly structure to adaptively combine two RBs.

2.2. Efficient Image Super-Resolution

In order to deploy SR models, several aspects should be also taken into consideration except restoration accuracy, such as the number of parameters, FLOPs, peak memory consumption and inference time. Recent methods towards EISR can be roughly divided into explicit [1, 17, 20, 29, 39] and implicit schemes [1, 16, 22, 40, 47]. The former reduces the model complexity via cutting down the width

and depth [29], recurrent structure [20, 39] and group convolution [1, 17]. The aforementioned strategies cause either accuracy loss or more extra overheads (*e.g.*, FLOPs). The latter implicit scheme focuses on sufficiently leveraging middle features as well as enhancing the discriminative ability, thus leading to lower complexity and better performance on the whole. For example, LapSRN [22] took advantage of layered pyramid features to reconstruct residuals of various resolution. MemNet [40] adopted gating mechanism to bridge deep features with shallow information. CARN [1] presented local and global cascading mechanism inspired by SRDenseNet [42] to boost up the representation power. IMDN [16] retained partial features as refined information and aggregated the distilled features via contrast-aware channel attention block. LatticeNet [47] applied a butterfly structure to dynamically combine two RBs. RFDN [32] enhances IMDB via feature distillation connection. DLSR [14] proposes a differentiable neural architecture search approach to find more powerful fusion blocks based on RFDB. It can be seen that feature fusion plays a pivotal role in recent advances. While theoretically efficient, it is sub-optimal due to relevant features residing in the memory until aggregation, leading to multiple times of memory consumption compared with simple topology counterparts without feature fusion.

2.3. Attention Mechanism in SR

Attention mechanism has been shown to be extremely powerful and applied in conjunction with CNN for various computer vision tasks, including SR [7, 16, 18, 32, 46, 51, 52]. It aims at guiding the network to focus on important signals and suppress unnecessary ones. Since the success of SENet [18] for image classification, various kinds of attention mechanisms have been applied to SR models. RCAN [51] first integrated channel attention (CA) into RB for SISR. RNAN [52] introduced local and non-local blocks to adaptively rescale the hierarchical features. SAN [7] used second-order feature statistics to generate more representative channel attention map. RFA [33] employed an enhanced spatial attention (ESA) block to obtain more sophisticated attention map. HAN [37] proposed layer attention and channel-spatial attention to model the holistic inter-dependencies among layers, channels, and positions. Recently, transformer-based methods [6, 30] have also been introduced to SR. SwinIR [30] applied shifted window mechanism to model long-range dependency, which can be interpreted as spatially varying convolution for capturing content-based interactions between image content and attention weights. They have achieved significant progress, but are not efficient enough for EISR due to multi-branch topology, multiple features residing in the memory and inefficient operations.

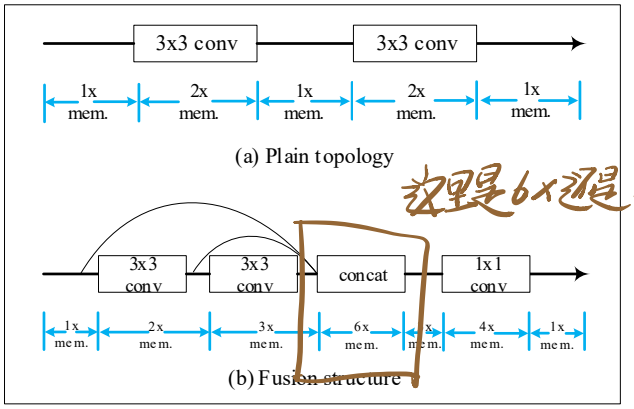


Figure 2. Peak memory consumption of plain topology and fusion structure during model inference. Suppose feature size remains the same after 3×3 convolution. Multiple features are fused in the concatenation node along the channel dimension and 1×1 convolution is used to reduce the final output feature size to the initial input feature size in (b). Activation layers are omitted for simplicity.

3. Proposed Method

3.1. Memory Analysis

We first introduce the memory analysis during inference which motivates the design of our network architecture. Memory consumption is an important factor for EISR model deployment. Generally, the memory consumption M at one node is composed of four parts: input feature memory M_{input}, output feature memory M_{output}, kept feature memory M_{kept} which is calculated before and used in the future nodes, network parameter memory M_{net}. The total memory M can be formulated as: $M = M_{input} + M_{output} + M_{kept} + M_{net}$. M_{net} is too small and thus negligible compared with feature memory. To compare the memory consumption of plain network topology and feature fusion schemes, we consider two typical structures of plain topology and feature fusion of EISR in Fig. 2 (suppose the input occupies 1x memory). For inplace ReLU layer, the input and output share the same memory chunk so M_{output} is zero, and we omit it for the simplicity of discussion. For a convolution layer with kernel size $C_{in} \times C_{out} \times K \times K$, M_{input} and M_{output} cannot be shared due to $C_{out} \times K \times K$ visits of every input location and Winograd [23]. Considering plain topology of a common 3×3 convolution in Fig. 2(a), the peak memory consumption in the convolution node is determined by $M_{input} + M_{output}$. If the feature size is not changed in the whole process, the peak memory consumption for this plain topology is around $2 \times C \times H \times W$, where C is the number of feature maps and H, W denote the height and width of each feature map, respectively. Stacking the same network topology sequen-

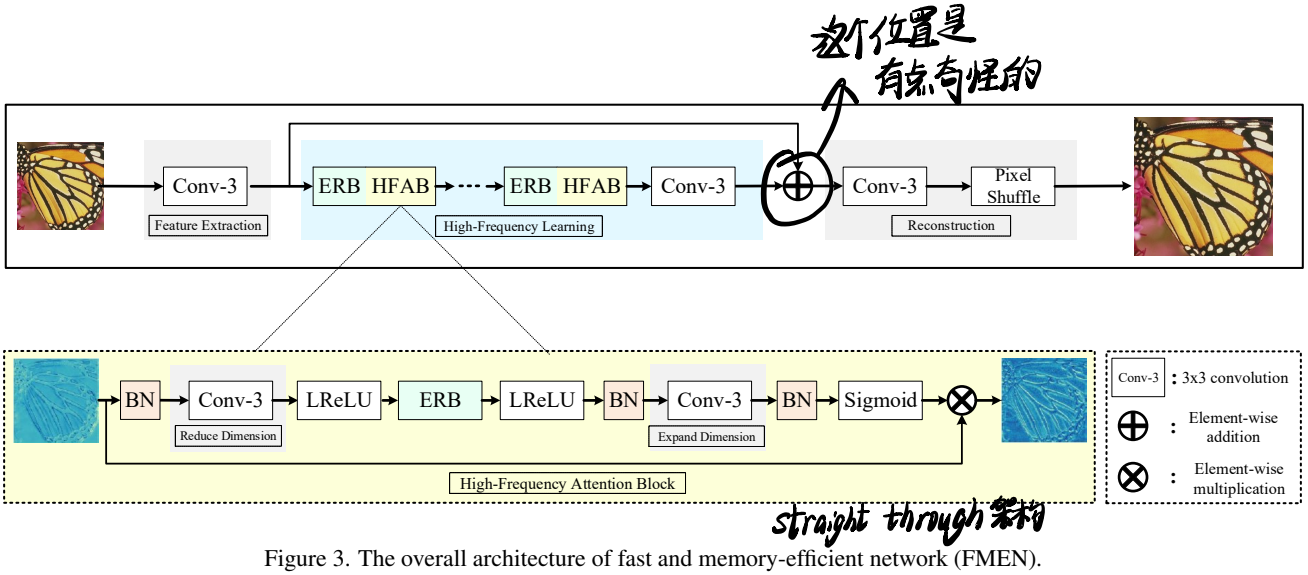


Figure 3. The overall architecture of fast and memory-efficient network (FMEN).

tially does not increase the peak memory consumption during inference.

As for the fusion structure, feature maps related to the fusion layer need to be kept until concatenation is finished. Taking the case of Fig. 2(b) as an example, there are three features occupying memory in the second 3×3 convolution layer: the input and output of this layer, the input of the first 3×3 convolution layer which will be used in the latter concatenation node, so the peak memory consumption for this layer is $3\times$ as the input. By the same logic, peak memory consumption doubles at the following concatenation node of three input features. In general, if there are N features of the same size ($C \times H \times W$) taking part in fusion, the memory consumption is raised at least to $2 \times N \times C \times H \times W$, since M_{input} and M_{output} are both $N \times C \times H \times W$ for the concatenation node. If global and local fusions are used simultaneously like in current lightweight architectures [1, 16, 32, 40], N_{local} input features, N_{local} concatenated output features and $N_{global} - 1$ kept features will occupy memory concurrently at the concatenation node of the last local fusion part, where N_{global} and N_{local} are the number of features taking part in the global and local fusions, respectively. Therefore, feature fusion generally increases peak memory consumption during inference in comparison with plain sequential topology.

3.2. Network Architecture

Applying sequential network topology to EISR is not a trivial task. One way is to directly adopt fully sequential architecture [10], and another way is to replace normal convolution layers with re-parameterizable building blocks [50] to expand optimization space during training. However, both often suffer the performance drop comparing with recent advanced fusion topology [32, 47].

In addition to memory consumption, inference time is another key aspect for EISR models. Based on the sequential network topology, we propose an enhanced residual block (ERB) for deep feature learning and an effective high-frequency attention block (HFAB) for feature enhancement, both of which not only reduce memory consumption but also accelerate the inference. The overall network architecture is shown in Fig. 3, which contains three main parts: feature extraction part, detail learning part and reconstruction part. The first and the last parts are kept the same as previous works [16, 31, 47, 50]. The second part is composed of alternating ERB and HFAB. The peak memory consumption of the entire model is reached at each element-wise multiplication node inside HFAB, where the memory is occupied by global residual, input feature of corresponding HFAB, the attention map and the output feature of corresponding HFAB, and is about $4 \times C \times H \times W$.

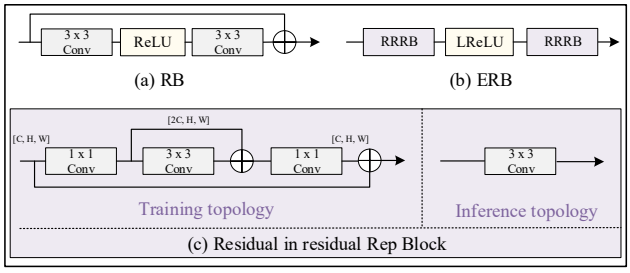


Figure 4. Architecture of vanilla RB proposed in EDSR [31] and our proposed ERB.

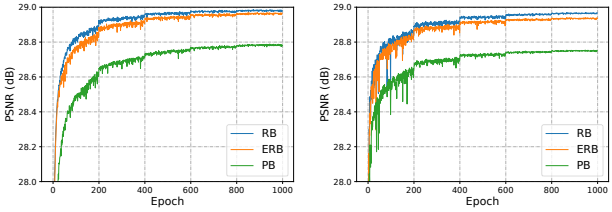
3.3. Enhanced Residual Block

Residual block (RB) proposed in [12] has been widely used in network models. It was customized and shown effective for SR in EDSR [31] and thus has been used in plenty of succeeding SR studies [1, 47, 51, 52]. We refer to RB as the version from EDSR in the following unless otherwise stated. However, the use of skip connection in-

troduces extra memory consumption of feature map size $C \times H \times W$, and lowers the inference speed due to additional memory access cost, which is shown experimentally that if the skip connection is removed in EDSR-baseline [31] the model runtime is reduced about 10% in Sec. 4.2. To inherit the merit of residual learning without introducing the aforementioned cost, we design ERB to replace RB. ERB is composed of one Leaky ReLU non-linearity and two residual in residual re-parameterization blocks (RRRB), which is inspired by RCAN [51] and RepVGG [8]. RRRB excavates the potential ability of sophisticated structure during optimization, while being equivalent to a single 3×3 convolution during inference. The network structure comparison between RB and ERB is illustrated in Fig. 4.

3.4. High-Frequency Attention Block

Recently, attention mechanism has been extensively studied in the SR literature. Based on the grain-size composition, it can be divided into channel attention [16, 18], spatial attention [33, 46], pixel attention [54], and layer attention [37]. Previous attention blocks [16,32,37] are multi-branch topology and contain inefficient operators, which cause extra memory consumption as discussed in Sec. 3.1 and slow the inference speed. Considering both aspects, we design a high-frequency attention block (HFAB) as shown in Fig. 3. The attention branch is responsible for assigning a scaling factor to every pixel, and high-frequency areas are expected to be assigned larger values since they mainly influence the restoration accuracy [13, 25]. We first reduce the channel dimension for efficiency by 3×3 convolution instead of 1×1 convolution. Afterwards ERB is applied to capture local interaction. Next, channel dimension increases to the original level and a sigmoid layer is used to restrict the value from 0 to 1. Finally, input features are recalibrated by multiplying the attention map in a pixel-wise manner. The motivation of the above steps is mainly from edge detection, where the linear combination of nearby pixels can be used to detect edges. The receptive field brought by convolutions is very limited, which means only the local-range dependency is modeled to determine the importance of every pixel. Thus, batch normalization (BN) is injected into the sequential layers to introduce global interaction, while being beneficial to the unsaturated area of the sigmoid function. Although previous works [31, 44] reported that BN could lead to some unexpected artifacts, we empirically observe that BN plays a role in restricting the pixel range diversity, which is in line with the high-frequency learning design, thus contributing to the performance improvement. During inference, we remove the skip connection of ERB inside HFAB and BN layers by merging the corresponding parameters into related convolution layers. HFAB only contains four highly optimized operators: 3×3 convolution, Leaky ReLU non-linearity, sigmoid and element-wise multiplication. HFAB avoids complex multi-branch topology, and thus ensures faster inference in comparison with ESA [33]. Rescaling features in the pixel level makes HFAB more powerful than CCA [16] in the channel level. The experiment details can be found in Sec. 4.4.



(a) EDSR-baseline [31] (b) FMEN

Figure 5. PSNR comparison of three blocks by applying them to EDSR-baseline [31] and FMEN: PB (remove skip connection in RB), RB [31] and proposed ERB. The performance is reported on DIV2K validation Set for x4 upscaling.

4. Experiments

4.1. Settings

Datasets and metrics. Following recent works [32, 47], we adopt widely used high-quality (2K resolution) DIV2K [41] and Flickr2K as training dataset. For testing, five standard benchmark datasets are used: Set5 [2], Set14 [48], B100 [34], Urban100 [15], and Manga109 [35]. The LR images are generated in the same way as [1, 16, 31, 47]. To keep consistence with existing methods [16, 17, 31, 40, 47], the SR results are evaluated on the luminance channel of transformed YCbCr space, using two common metrics called peak signal-to-noise ratio (PSNR) and structure similarity index (SSIM) [45]. Following AIM 2020 [49], maximum GPU memory consumption is tested using PyTorch function $torch.cuda.max\_memory\_allocated$ on the LR image of size 256×256.

Implementation details. In each training batch, 64 cropped 64×64 LR RGB patches augmented by random flipping and rotation are input to the network. L1 loss is adopted to train our model. The learning rate is initialized as 5×10^{-4} and decreases half per 6×10^5 iterations for total 3×10^6 iterations. The number of convolution kernels in the trunk of ERB is set to 64 and 64, and the first convolution in HFAB reduces the channel dimension to 32. The number of ERB and HFAB pairs is set to 5 to achieve comparable performance as LatticeNet [47]. Parameters of our model are initialized using the method proposed by He *et al.* [11] and optimized by ADAM optimizer [21] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. Our model is implemented using the PyTorch framework on an Nvidia 1080Ti GPU.

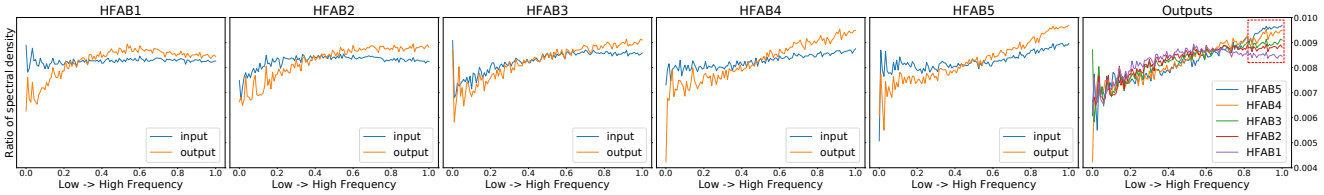


Figure 6. Frequency analysis of input and output in each HFAB.

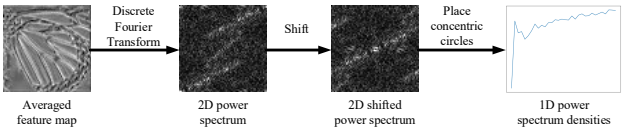


Figure 7. Frequency analysis pipeline [29, 40].

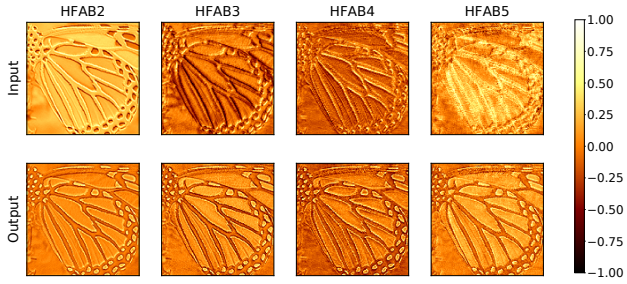


Figure 8. Average feature maps of the four fine grained HFABs. Since the absolute values of outputs are always smaller than the inputs', we normalize them for better visualization.

Table 1. Inference time comparison between RB and ERB. It is the average of 10 runs on DIV2K validation set for x4 upscaling.

	MemNet [40]	EDSR-baseline [31]	FMEN
RB	231.7ms	99.8ms	46.1ms
ERB	214.5ms (↓7.5%)	90.6ms (↓9.2%)	40.4ms (↓12.4%)

4.2. Effectiveness of ERB

ERB is proposed to combine the advantages of sequential topology and residual learning. To validate its effectiveness, we compare three blocks: plain block (PB) by removing the skip connection in RB, RB and ERB. They are of the same number of parameters and we apply them separately to EDSR-baseline [31] and FMEN. In detail, original EDSR-baseline [31] adopts RBs to construct the network, we replace it with PBs or ERBs. For FMEN, we replace ERBs with RBs or PBs. From Fig. 5, we can see that ERB architecture achieves comparable performance as RB architecture, while PB architecture behaves much worse (nearly 0.2dB drop). During inference, ERB can be reducible to PB thus enjoys the efficiency of sequential topology. We also report the inference speed comparison in Tab. 1. ERB decreases around 10% inference time by avoiding memory access cost (MAC).

Table 2. Investigation of different attention mechanisms. All models are validated on Set5 and Urban100 for $\times 4$ upscaling in 300 epochs. We also report the average inference time on DIV2K validation set for $\times 4$ upscaling on an Nvidia 1080Ti GPU. The last column shows time increase rate compared with Baseline.

Model	Params	Set5	Urban100	Inference time	Increase rate
Baseline	779K	32.03	25.89	45.25ms	0.0%
Baseline+CCA [16]	782K	32.11	25.93	50.55ms	11.7%
Baseline+ESA [32]	784K	32.12	25.95	52.11ms	15.2%
FMEN	769K	**32.17**	**26.04**	46.13ms	1.9%

4.3. Frequency Analysis

To figure out whether HFAB focuses on high-frequency areas as we expected, we analyze the feature frequency distribution before and after each HFAB. Inspired by MemNet [40] and SRFBN [29], we center the power spectra of the average feature maps and estimate spectral densities for a continuous set of frequency by placing concentric circles. We divide spectral density of each frequency by the sum of spectral densities to intuitively observe what percentage low frequency and high frequency account for. The process of calculating power spectrum density (PSD) is displayed in Fig. 7. Based on it, we show the frequency analysis results in Fig. 6, from which we have three main observations. First, processed by HFAB, low-frequency signals of input features are suppressed and high-frequency signals are strengthened. Second, the inputs (processed by ERBs in the trunk branch) of HFAB slowly focus on high-frequency signals while our HFAB can immediately recalibrate interest areas, which explains why our design has superiority over the stacked RB architecture. Third, the separability of frequency can be further enhanced by posterior HFAB (see the rightmost figure in Fig. 6).

More intuitively, we plot the inputs and outputs of the four fine grained HFABs (The first HFAB seems to adjust middle-frequency signals in Fig. 6). The visualization shown in Fig. 8 is an efficient complement to the conclusions drawn from Fig. 6, which provides a deeper insight that the enhanced areas are almost edges and other details.

怪不得x大Flops小

Table 3. Quantitative results on benchmark datasets. Red indicates the best and blue indicates the second best. The image size of HR is set to 1280 × 720 to calculate the Mult-Adds.

Method	Scale	Params	Mult-Adds	Set5	Set14	BSD100	Urban100	Manga109
				PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Bicubic		-	-	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	26.88 / 0.8403	30.80 / 0.9339
SRCNN [9]		8K	52.7G	36.66 / 0.9542	32.45 / 0.9067	31.36 / 0.8879	29.50 / 0.8946	35.60 / 0.9663
FSRCNN [10]		13K	6.0G	37.00 / 0.9558	32.63 / 0.9088	31.53 / 0.8920	29.88 / 0.9020	36.67 / 0.9710
VDSR [19]		666K	612.6G	37.53 / 0.9587	33.03 / 0.9124	31.90 / 0.8960	30.76 / 0.9140	37.22 / 0.9750
DRCN [20]		1774K	9,788.7G	37.63 / 0.9588	33.04 / 0.9118	31.85 / 0.8942	30.75 / 0.9133	37.55 / 0.9732
LapSRN [22]		253K	104.9G	37.52 / 0.9591	32.99 / 0.9124	31.80 / 0.8952	30.41 / 0.9103	37.27 / 0.9740
DRRN [39]		298K	6,796.9G	37.74 / 0.9591	33.23 / 0.9136	32.05 / 0.8973	31.23 / 0.9188	37.88 / 0.9749
MemNet [40]	x2	678K	623.9G	37.78 / 0.9597	33.28 / 0.9142	32.08 / 0.8978	31.31 / 0.9195	37.72 / 0.9740
IDN [17]		579K	133.0G	37.83 / 0.9600	33.30 / 0.9148	32.08 / 0.8985	31.27 / 0.9196	38.01 / 0.9749
EDSR-baseline [31]		1370K	316.2G	37.91 / 0.9602	33.53 / 0.9172	32.15 / 0.8995	31.99 / 0.9270	38.40 / 0.9766
CARN [1]		1592K	222.8G	37.76 / 0.9590	33.52 / 0.9166	32.09 / 0.8978	31.92 / 0.9256	38.36 / 0.9765
IMDN [16]		694K	158.8G	38.00 / 0.9605	33.63 / 0.9177	32.19 / 0.8996	32.17 / 0.9283	38.88 / 0.9774
LatticeNet [47]		756K	169.5G	38.15 / 0.9610	33.78 / 0.9193	32.25 / 0.9005	32.43 / 0.9302	- / -
RFDN [32]		534K	123.0G	38.05 / 0.9606	33.68 / 0.9184	32.16 / 0.8994	32.12 / 0.9278	38.88 / 0.9773
FMEN		748K	172.0G	38.10 / 0.9609	33.75 / 0.9192	32.26 / 0.9007	32.41 / 0.9311	38.95 / 0.9778
Bicubic		-	-	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	24.46 / 0.7349	26.95 / 0.8556
SRCNN [9]		8K	52.7G	32.75 / 0.9090	29.30 / 0.8215	28.41 / 0.7863	26.24 / 0.7989	30.48 / 0.9117
FSRCNN [10]		13K	5.0G	33.18 / 0.9140	29.37 / 0.8240	28.53 / 0.7910	26.43 / 0.8080	31.10 / 0.9210
VDSR [19]		666K	612.6G	33.66 / 0.9213	29.77 / 0.8314	28.82 / 0.7976	27.14 / 0.8279	32.01 / 0.9340
DRCN [20]		1774K	9,788.7G	33.82 / 0.9226	29.76 / 0.8311	28.80 / 0.7963	27.15 / 0.8276	32.24 / 0.9343
LapSRN [22]		290K	115.0G	33.81 / 0.9220	29.79 / 0.8325	28.82 / 0.7980	27.07 / 0.8275	32.21 / 0.9350
DRRN [39]		298K	6,796.9G	34.03 / 0.9244	29.96 / 0.8349	28.95 / 0.8004	27.53 / 0.8378	32.71 / 0.9379
MemNet [40]	x3	678K	623.9G	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	27.56 / 0.8376	32.51 / 0.9369
IDN [17]		588K	60.1G	34.14 / 0.9259	30.13 / 0.8383	28.98 / 0.8026	27.86 / 0.8463	33.11 / 0.9416
EDSR-baseline [31]		1554K	160.4G	34.28 / 0.9263	30.24 / 0.8405	29.06 / 0.8044	28.00 / 0.8493	33.37 / 0.9432
CARN [1]		1592K	118.9G	34.29 / 0.9255	30.29 / 0.8407	29.06 / 0.8034	28.06 / 0.8493	33.50 / 0.9440
IMDN [16]		703K	71.5G	34.36 / 0.9270	30.32 / 0.8417	29.09 / 0.8046	28.17 / 0.8519	33.61 / 0.9445
LatticeNet [47]		765K	76.3G	34.53 / 0.9281	30.39 / 0.8424	29.15 / 0.8059	28.33 / 0.8538	- / -
RFDN [32]		541K	55.4G	34.41 / 0.9273	30.34 / 0.8420	29.09 / 0.8050	28.21 / 0.8525	33.67 / 0.9449
FMEN		757K	77.2G	34.45 / 0.9275	30.40 / 0.8435	29.17 / 0.8063	28.33 / 0.8562	33.86 / 0.9462
Bicubic		-	-	28.42 / 0.8104	26.00 / 0.7027	25.96 / 0.6675	23.14 / 0.6577	24.89 / 0.7866
SRCNN [9]		8K	52.7G	30.48 / 0.8626	27.50 / 0.7513	26.90 / 0.7101	24.52 / 0.7221	27.58 / 0.8555
FSRCNN [10]		13K	4.6G	30.72 / 0.8660	27.61 / 0.7550	26.98 / 0.7150	24.62 / 0.7280	27.90 / 0.8610
VDSR [19]		666K	612.6G	31.35 / 0.8838	28.01 / 0.7674	27.29 / 0.7251	25.18 / 0.7524	28.83 / 0.8870
DRCN [20]		1774K	9,788.7G	31.53 / 0.8854	28.02 / 0.7670	27.23 / 0.7233	25.14 / 0.7510	28.93 / 0.8854
LapSRN [22]		543K	139.3G	31.54 / 0.8852	28.09 / 0.7700	27.32 / 0.7275	25.21 / 0.7562	29.09 / 0.8900
DRRN [39]		298K	6,796.9G	31.68 / 0.8888	28.21 / 0.7720	27.38 / 0.7284	25.44 / 0.7638	29.45 / 0.8946
MemNet [40]	x4	678K	623.9G	31.74 / 0.8893	28.26 / 0.7723	27.40 / 0.7281	25.50 / 0.7630	29.42 / 0.8942
IDN [17]		600K	34.5G	31.93 / 0.8923	28.45 / 0.7781	27.48 / 0.7326	25.81 / 0.7766	30.04 / 0.9026
EDSR-baseline [31]		1518K	114.2G	31.98 / 0.8927	28.55 / 0.7805	27.54 / 0.7348	25.90 / 0.7809	30.24 / 0.9053
CARN [1]		1592K	90.9G	32.13 / 0.8937	28.60 / 0.7806	27.58 / 0.7349	26.07 / 0.7837	30.47 / 0.9084
IMDN [16]		715K	40.9G	32.21 / 0.8948	28.58 / 0.7811	27.56 / 0.7353	26.04 / 0.7838	30.45 / 0.9075
LatticeNet [47]		777K	43.6G	32.30 / 0.8962	28.68 / 0.7830	27.62 / 0.7367	26.25 / 0.7873	- / -
RFDN [32]		550K	31.6G	32.24 / 0.8952	28.61 / 0.7819	27.57 / 0.7360	26.11 / 0.7858	30.58 / 0.9089
FMEN		769K	44.2G	32.24 / 0.8955	28.70 / 0.7839	27.63 / 0.7379	26.28 / 0.7908	30.70 / 0.9107

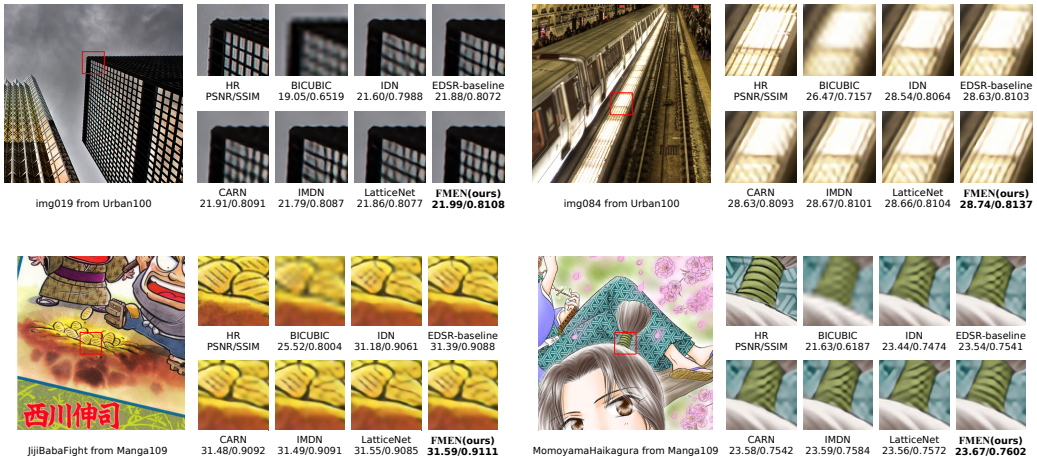


Figure 9. Visual results on Urban100 and Manga109 for ×4 upscaling.

Table 4. Efficiency comparison on DIV2K validation set for x4 upscaling, with PyTorch 1.4.0, CUDA Toolkit 10.0.130, cuDNN 7.6.5, on an NVIDIA 1080Ti GPU. Red indicates the best and blue indicates the second best.

Method	Params	FLOPs	Runtime	Memory	Activations	Convs	DIV2K val.
IMDN [16]	894K	58.53G	50.86ms	471.76M	154.14M	43	29.13
E-RFDN [32]	433K	27.1G	41.97ms	788.13M	112.03M	64	29.04
ByteESR	317K	19.7G	27.11ms	377.91M	80.05M	39	29.00
NEESR	272K	16.86G	29.97ms	575.99M	79.59M	59	29.01
Super	326K	20.06G	32.09ms	663.07M	93.82M	59	29.00
MegSR	290K	17.7G	32.59ms	640.63M	91.72M	64	29.00
rainbow	276K	17.98G	34.1ms	309.23M	92.8M	59	29.01
FMEN-S(ours)	341K	22.28G	28.07ms	204.6M	72.09M	34	29.00

4.4. Comparison with Other Attention Mechanisms

As we all know, the attention mechanism has been shown useful to boost the performance of EISR networks. In this subsection, we compare recent attention mechanisms with our scheme. As discussed in [16], depth is most related to the execution speed, so we construct a baseline with 15 ERBs in the detail learning part, which has the similar depth and number of parameters as ours. Since there are 5 HFABs in our network, we insert an attention block every 3 ERBs into the baseline so that attention mechanism is applied 5 times as well. Specifically, we investigate contrast-aware channel attention (CCA) [16, 47], and enhanced spatial attention (ESA) [33]. The inserted mode keeps the same with FMEN. Although high learning rate is more beneficial for FMEN, we set the learning rate to 1×10^{-4} to suit for other attention models. Table. 2 shows that the performance of baseline can be improved by embedding current attention blocks, but the gain is relatively lower than that of our scheme. Moreover, with the same number of attention blocks, competing attention mechanisms cause larger time overhead due to MAC brought by multi-branch topology [16, 47] and inefficient operations such as 7×7 convolution [32], while our scheme is more friendly for network inference without time-consuming operations.

4.5. Comparison with State-of-the-art Methods

In this section, we compare our method with other lightweight SR models: SRCNN [9], FSRCNN [10], VDSR [19], DRCN [20], LapSRN [22], DRRN [39], MemNet [40], IDN [17], EDSR-baseline [31], CARN [1], IMDN [16], LatticeNet [47] and RFDN [32]. The quantitative comparisons for x2, x3, x4 upscaling on five publicly available SR benchmark datasets are shown in Tab. 3. FMEN achieves comparable PSNR and SSIM with the most competitive EISR method, LatticeNet, but gains clear advantage in terms of runtime (46ms vs. 68ms) and memory consumption (68M vs. 225M). Visual comparisons are illustrated in Fig. 9. Our method yields more visually pleasant patterns in the selected Urban100 and Manga109 images, compared with other state-of-the-art methods.

4.6. NTIRE 2022 Challenge on Efficient Super-Resolution

To participate in this competition, we reduce the number of convolution kernels to 50 in every convolution layer except for HFAB to maintain the PSNR of 29.00dB on DIV2K validation set. We denote this model as *FMEN-S*. It is worth mentioning that the baseline of the competition [28], RFDN, reduces the number of RFDB from 6 to 4 and the number of parameters decreases from 550K to 433K, and it is denoted as *E-RFDN* here [32]. We follow the official evaluation setting and report the number of parameters, FLOPs, runtime, peak memory consumption, activations and number of convolution in Tab. 4. Recent advanced EISR methods IMDN [16], E-RFDN [32] and the top methods in [28] are included for comparison. Our method achieves the lowest memory consumption and the second shortest runtime, while maintaining comparable restoration accuracy. Specifically, compared with AIM 2020 winner solution E-RFDN [32], our model can decrease 21.2% parameters, 17.8% FLOPs, 33.1% runtime, 74% peak memory consumption and 35.7% activations, with only 0.04dB PSNR drop. Compared with other participants in NTIRE 2022 challenge on efficient super-resolution [28], our model achieves the best memory consumption, number of activations and convolutions, and the second best inference speed. In contrast to the theoretical metrics, the above metrics are more important in the practical use.

5. Conclusions

In this paper, we analyze the factors which influence runtime and memory consumption of current EISR models, and design a fast and memory-efficient network (FMEN) with efficient sequential operators and attention mechanism. FMEN is mainly composed of two basic blocks: enhanced residual block (ERB) and high-frequency attention block (HFAB). ERB takes advantage of RB but more friendly for deployment. HFAB is more powerful and lightweight than conventional attention blocks. Our method gains substantial improvement of runtime and memory consumption, while maintaining comparable reconstruction performance.

References

[1] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV (10)*, volume 11214 of *Lecture Notes in Computer Science*, pages 256–272. Springer, 2018. 1, 2, 3, 4, 5, 7, 8

[2] Marco Bevilacqua, A. Roumy, Christine Guillemot, and Marie-Line Alberi-Morel. Low-complexity single image super-resolution based on nonnegative neighbor embedding. 09 2012. 5

[3] Y. Bhalgat, Y. Zhang, J. Lin, and F. Porikli. Structured convolutions for efficient neural network design. 2020. 1

[4] Kelvin C. K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond, 2021. 2

[5] S. K. Chao, Z. Wang, Y. Xing, and G. Cheng. Directional pruning of deep neural networks. 2020. 1

[6] 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, 2021. 3

[7] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[8] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again, 2021. 2, 5

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV (4)*, volume 8692 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 2014. 1, 2, 7, 8

[10] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV (2)*, volume 9906 of *Lecture Notes in Computer Science*, pages 391–407. Springer, 2016. 4, 7, 8

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 2015. 5

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2, 4

[13] Y. He, X. Dong, G. Kang, Y. Fu, and Y. Yang. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE Transactions on Cybernetics*, PP(99):1–11, 2019. 5

[14] Han Huang, Li Shen, Chaoyang He, Weisheng Dong, Haozhi Huang, and Guangming Shi. Lightweight image super-resolution with hierarchical and differentiable neural architecture search, 2021. 3

[15] Jia Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 5

[16] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACM Multimedia*, pages 2024–2032. ACM, 2019. 1, 2, 3, 4, 5, 6, 7, 8

[17] Zheng Hui, Xiumei Wang, and Xinbo Gao. Fast and accurate single image super-resolution via information distillation network. In *CVPR*, pages 723–731. IEEE Computer Society, 2018. 1, 2, 3, 5, 7, 8

[18] Jie, Hu, Li, Shen, Samuel, Albanie, Gang, Sun, Enhua, and Wu. Squeeze-and-excitation networks. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 3, 5

[19] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654. IEEE Computer Society, 2016. 2, 7, 8

[20] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, pages 1637–1645. IEEE Computer Society, 2016. 2, 3, 7, 8

[21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computer ence*, 2014. 5

[22] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, pages 5835–5843. IEEE Computer Society, 2017. 2, 3, 7, 8

[23] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. In *CVPR*, pages 4013–4021, 2016. 3

[24] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, and Zehan and Wang. Photo-realistic single image super-resolution using a generative adversarial network. 2017. 2

[25] W. Lee, J. Lee, D. Kim, and B. Ham. Learning with privileged information for efficient image super-resolution. 2020. 5

[26] Y. Li, S. Gu, K. Zhang, L Van Gool, and R. Timofte. Dhp: Differentiable meta pruning via hypernetworks. 2020. 1

[27] Y. Li, W. Li, M. Danelljan, K. Zhang, and R. Timofte. The heterogeneity hypothesis: Finding layer-wise dissimilated network architecture. 2020. 1

[28] Yawei Li, Kai Zhang, Luc Van Gool, Radu Timofte, et al. Ntire 2022 challenge on efficient super-resolution: Methods and results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2022. 1, 8

[29] Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu. Feedback network for image super-resolution. In *CVPR*, 2019. 1, 2, 3, 6

[30] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *IEEE International Conference on Computer Vision Workshops*, 2021. 3

[31] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. 2017. 1, 2, 4, 5, 6, 7, 8

[32] J. Liu, J. Tang, and G. Wu. Residual feature distillation network for lightweight image super-resolution. 2020. 1, 2, 3, 4, 5, 6, 7, 8

[33] Jie Liu, Wenjie Zhang, Yuting Tang, Jie Tang, and Gangshan Wu. Residual feature aggregation network for image super-resolution. pages 2356–2365, 06 2020. 2, 3, 5, 8

[34] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision*, 2002. 5

[35] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017. 5

[36] H. Mobahi, M. Farajtabar, and P. L. Bartlett. Self-distillation amplifies regularization in hilbert space. 2020. 1

[37] B. Niu, W. Wen, W. Ren, X. Zhang, L. Yang, S. Wang, K. Zhang, X. Cao, and H Shen. Single image super-resolution via a holistic attention network. 2020. 3, 5

[38] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. 2016. 2

[39] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *CVPR*, pages 2790–2798. IEEE Computer Society, 2017. 2, 3, 7, 8

[40] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, pages 4549–4557. IEEE Computer Society, 2017. 1, 2, 3, 4, 5, 6, 7, 8

[41] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming Hsuan Yang, and Qi Guo. Ntire 2017 challenge on single image super-resolution: Methods and results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017. 5

[42] T. Tong, G. Li, X. Liu, and Q. Gao. Image super-resolution using dense skip connections. In *IEEE International Conference on Computer Vision*, 2017. 3

[43] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. 2020. 1

[44] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Chen Change Loy, Yu Qiao, and Xiaoou Tang. Esrgan: Enhanced super-resolution generative adversarial networks, 2018. 2, 5

[45] Zhou Wang, Alan Bovik, Hamid Sheikh, and Eero Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13, 09 2004. 5

[46] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, pages 3–19, 2018. 3, 5

[47] Luo Xiaotong, Xie Yuan, Zhang Yulun, Qu Yanyun, Li Cuihua, and Fu Yun. Latticenet: Towards lightweight image super-resolution with lattice block. 2020. 1, 2, 3, 4, 5, 7, 8

[48] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International Conference on Curves and Surfaces*, 2010. 5

[49] 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 *European Conference on Computer Vision*, pages 5–40. Springer, 2020. 1, 5

[50] Xindong Zhang, Hui Zeng, and Lei Zhang. Edge-oriented convolution block for real-time super resolution on mobile devices. 2021. 2, 4

[51] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. 2018. 1, 2, 3, 4, 5

[52] Yulun Zhang, Kunpeng Li, Kai Li, Bineng Zhong, and Yun Fu. Residual non-local attention networks for image restoration. 2019. 2, 3, 4

[53] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. 2018. 2

[54] Hengyuan Zhao, Xiangtao Kong, Jingwen He, Yu Qiao, and Chao Dong. Efficient image super-resolution using pixel attention. In *ECCV*, pages 56–72. Springer, 2020. 5