Edge-oriented Convolution Block for Real-time Super Resolution on Mobile Devices

Xindong Zhang*
The Hong Kong Polytechnic University
DAMO Academy, Alibaba Group
csxdzhang@comp.polyu.edu.hk

Hui Zeng*
The Hong Kong Polytechnic University
DAMO Academy, Alibaba Group
cshzeng@comp.polyu.edu.hk

Lei Zhang†
The Hong Kong Polytechnic University
DAMO Academy, Alibaba Group
cslzhang@comp.polyu.edu.hk

ABSTRACT

Efficient and light-weight super resolution (SR) is highly demanded in practical applications. However, most of the existing studies focusing on reducing the number of model parameters and FLOPs may not necessarily lead to faster running speed on mobile devices. In this work, we propose a re-parameterizable building block, namely Edge-oriented Convolution Block (ECB), for efficient SR design. In the training stage, the ECB extracts features in multiple paths, including a normal 3×3 convolution, a channel expanding-and-squeezing convolution, and 1st-order and 2nd-order spatial derivatives from intermediate features. In the inference stage, the multiple operations can be merged into one single 3×3 convolution. ECB can be regarded as a drop-in replacement to improve the performance of normal 3×3 convolution without introducing any additional cost in the inference stage. We then propose an extremely efficient SR network for mobile devices based on ECB, namely ECBSR. Extensive experiments across five benchmark datasets demonstrate the effectiveness and efficiency of ECB and ECBSR. Our ECBSR achieves comparable PSNR/SSIM performance to state-of-the-art light-weight SR models, while it can super resolve images from 270p/540p to 1080p in real-time on commodity mobile devices, e.g., Snapdragon 865 SOC and Dimensity 1000+ SOC. The source code can be found at https://github.com/xindongzhang/ECBSR.

CCS CONCEPTS

• **Computing methodologies** → **Computational photography**; **Reconstruction**.

KEYWORDS

image super-resolution; real-time network; edge-oriented convolution; mobile vision

ACM Reference Format:
Xindong Zhang, Hui Zeng, and Lei Zhang. 2021. Edge-oriented Convolution Block for Real-time Super Resolution on Mobile Devices. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3474085.3475291

*Both authors contributed equally to this research.
†Corresponding author

MM '21, October 20–24, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3475291

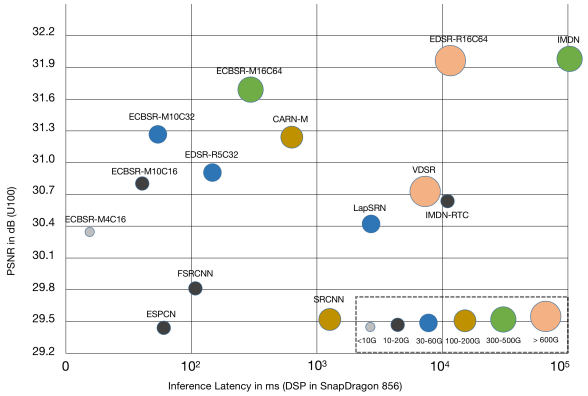


Figure 1: Illustration of PSNR on Urban100 dataset, FLOPs range (marked by color cast) and inference latency of different SISR models performing 540p to 1080p upscaling. All models are quantized and executed on DSP hardware of SnapDragon 865.

1 INTRODUCTION

Singe Image super resolution (SISR) aims at reproducing high-resolution (HR) outputs from their degraded low resolution (LR) counterparts. In recent years, deep convolutional neural network (DCNN) [25] based SR models [5, 12, 13, 30, 39, 40, 44, 48, 57] have become prevalent for their strong capability in recovering or generating [26, 49] image high-frequency details, showing promising practical value in image and video restoration, transition and display. However, most of existing SR models tend to employ very deep and complicated network topology for reproducing more details. As a result, the required heavy computational cost and memory consumption make it hard to deploy these SR models in many real-world applications with resource-limited edge and mobile devices.

In order to deploy SR models on resource-limited devices, how to improve their efficiency has attracted increasing attentions recently [55]. Model parameters reduction [22, 43] and FLOPs reduction [1, 7, 17, 31, 47, 55] are currently the two main streams towards efficient and light-weight SR design. The former strategy usually employs weight sharing strategy to reduce the model parameters, but it does

not change the computational complexity of network. The latter strategy focuses on introducing low-FLOPs (e.g., group convolution, depth-wise convolution, and element-wise operation) and FLOPs-free (e.g., feature splitting, concatenation and shuffling / reshaping) operations to reduce the consumption of FLOPs while maintaining competitive capacity to those large SR models. However, it has been shown recently that fewer number of parameters and FLOPs do not necessarily lead to faster running speed [55]. In addition, most of existing studies on efficient SR design are evaluated on GPU servers which cannot reflect their running speed on mobile devices.

It is more challenging to design light-weight and efficient SR model on commodity mobile devices due to the very limited hardware resources on mobile SOC, e.g., fairly lower bandwidth and computing capacity compared to those of GPU server. Fig. 1 shows the hardware performance of various SR models performing 540p to 1080p upscaling on the DSP hardware of SnapDragon 865, where all models conduct inference in 8-bit arithmetic. As can be seen, most of the compared SR models obtaining reasonable PSNR index are far from real-time in mobile hardware validation, and models with fewer FLOPs may have even larger latency. Although some tiny SR models such as FSRCNN [13] and ESPCN [40] can reach nearly real-time speed, their SR performance measured by PSNR is quite limited. It is highly desirable to design efficient SR model that can achieve real-time SR on mobile devices while maintaining reasonable SR quality.

In this work, we make an attempt to build super efficient SR model targets for mobile devices. We first choose a neat topology with deployable-friendly operators to ensure high inference speed. Although multi-branch topology can improve the SR performance, the increased network fragments severely decrease the inference speed on mobile devices. To boost the SR performance while maintaining high efficiency, we propose to employ the re-parameterization technique into our SR model design. By explicitly employing a linear combination of multiple branches in the training stage and folding them back to a normal 3x3 convolution in the inference stage, the re-parameterization strategy has proven its effectiveness on several high-level vision tasks [9–11, 53]. However, straightly borrowing their network blocks for SR task brings little improvement. To address this problem, we design a more effective re-parameterization block, namely Edge-oriented Convolution Block (ECB) for the SR task. Specifically, our ECB consists of four types of carefully designed operators: normal 3×3 convolution, channel expanding-and-squeezing convolution, extracting 1st-order and 2nd-order spatial derivatives from intermediate features. Such a design can more effectively extract edge and texture information which is important for SR task. As shown in Fig. 1, our ECB based SR model (ECBSR) can not only achieve high SR quality but also maintain high inference speed.

Our contributions can be summarized as follows:

1) We, for the first time, investigate the structure re-parameterizable technique for SR task and propose an Edge-oriented Convolution Block (ECB). ECB can be used to improve the SR performance of any SR model without introducing any extra burden for inference.
2) Based on ECB, we further design a super efficient and light-weight SR model, namely ECBSR, for real-time SISR on mobile devices.

Extensive experiments and comparisons validate the effectiveness of our proposed ECB and ECBSR. ECBSR can upscale images from 270p/540p to 1080p at real-time on mobile devices while preserving good SR quality.

2 RELATED WORK

Efficient SR Network Design. Existing efficient SR network design methods focus mainly on reducing the number of parameters and FLOPs. Kim et al. [21] proposed a very deep SR (VDSR) network via residual learning and introduced a deep recursive convolutional network (DRCN) [22] to reduce the number of parameters. Tai et al. [43] improved DRCN by combining recursive and residual learning to achieve better performance with fewer parameters. To reduce the number of FLOPs, Ahn et al. [1] proposed an efficient cascading residual network (CARN) with group convolution. Hui et al. [17] proposed an information distillation network (IDN) to compress the number of filters per layer. They then extended IDN to information multidistillation network (IMDN) [16] and won the AIM 2019 constrained image SR challenge [56]. Liu et al. [31] further improved IMDN to residual feature distillation block (RFDB) and won the AIM 2020 [55] SR challenge. In [7, 8], authors used FLOPs as a constraint of latency objective to search for a light-weight and effective SR network. However, fewer parameters and FLOPs do not necessarily mean higher efficiency, especially on the mobile devices. In this work, we discuss the factors that affect the efficiency of SR models on mobile devices and introduce a mobile-efficient SR model.

Computation Reduction. There are also some attempts trying to reduce the computational cost while preserving SR performance. Low-bit quantization [33, 51] represents features and weights of SR models in quantized format, which significantly reduces the model size and computational cost. However, quantization of SR models often can hardly maintain the SR quality because of the precision requirement of pixel-wise prediction. Model pruning is also an effective method for computation reduction. DHP [28] uses a differentiable prunning method via hypernetworks for automatic network pruning, and shows its effectiveness on SR task. Some researchers seek to replace the multiplication with cheaper operations. AdderSR [41] employs addition operation to replace multiplication and GhostSR [35] uses shift operator to generate redundant features. However, these special operations, such as binary/tenary convolution, fully adder convolution, and shift operation, require customized hardware optimization to achieve high inference speed, which is infeasible for most commodity mobile devices only supporting 8/16/32 bit arithmetic calculation and limited operations. To this end, our proposed ECB is friendly to most commodity mobile devices and it can be used to improve SR inference performance at no additional cost.

Re-parameterization. Recently, several studies on re-parameterization have shown their effectiveness on high level vision tasks such as image classification, object detection and semantic segmentation [2, 9–11, 53]. Arora et. al [2] demonstrated that the re-parameterization of FC layer can accelerate the training of network as depth of network increases. DiracNet [53] trains a plain CNN with comparable performance to ResNet series. ACNet[9] adopts asymmetric convolution to strengthen the normal convolution,

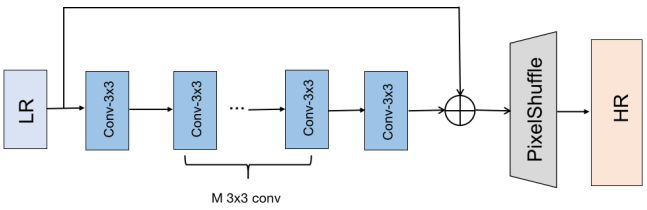
which can be viewed as another form of structural re-parameterization. RepVGG [11] de-couples a normal 3×3 convolution into multi-branch block consisting of identity mapping, 1×1 convolution and 3×3 convolution, and boost the performance of traditional VGG to the level of ResNet series [14] on several high level vision tasks. Diverse Branch Block [10] exploites different scales and complexities of features to enrich the representative capacity of normal convolution. Previous re-parameterization blocks have been validated on high level tasks, however it is not effective to straightly employ these blocks in SR task according to our experiments.

In this paper, we propose ECB for the SR task. ECB is a re-parameterization block which incorporates domain knowledge from classical directional and edge filters [3, 29, 36, 58]. Different from previous researches that explicitly extract the edge information using additional local filters, we implicitly insert learnable edge-aware filter groups in the manner of re-parameterization, which does not introduce additional cost in the inference stage.

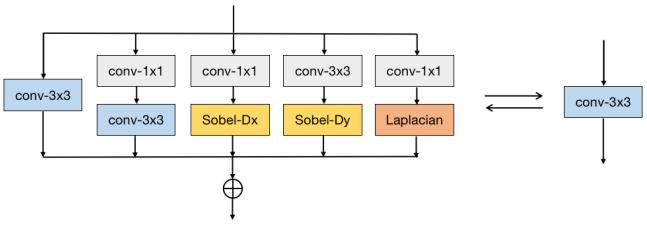
3 METHODOLOGY

We first design a neat topology with mobile-friendly operations as the base model, then propose a re-parameterizable edge-oriented convolution block to improve SR performance.

3.1 Base model



(a) Base topology.



(b) Edge-oriented convolution block (ECB).

Figure 2: Illustration of (a) base topology and (b) Edge-oriented Convolution Block (ECB). In the training stage, the ECB employs multiple branches, which can be merged into one normal convolution layer in the inference stage.

To ensure high inference speed and cross-device deployment on commodity mobile devices, we carefully consider the limited computation and memory resource on mobile devices and deliberately choose a neat topology consisting of the most basic operations as the base model. The overall architecture of the base model is shown in Fig. 2(a).

Neat Topology. Although complicated topologies such as multiple branches [1, 16, 17, 31] and dense connections [46, 49] can enrich the feature representation without introducing many FLOPs, such topologies result in much higher memory access cost (MAC) and sacrifice the parallelism degree, which severely reduces the inference speed. The situation is even worse on mobile devices because of the low bandwidth of DDR[1]. For one example, FSRCNN [13] using a plain topology has slightly higher FLOPs than IMDN-RTC [16] using a complicated topology, but runs about two-order faster on SnapDragon 865 for 540p to 1080p upscaling as shown in Fig. 1. Taking the limited bandwidth into consideration, we choose a nearly plain topology as the base model and employ only one skip connection in the three-channel image space (rather than the high-dimensional feature space) to keep the MAC of our model as low as possible.

Basic Operations. Unlike GPU servers, the operations well optimized by DSP/GPU/NPU on mobile devices are currently quite limited and vary from device to device. Unsupported operations have to be processed on CPU, not only having very low processing speed but also introducing additional MAC. We thus only adopt M 3×3 normal convolutions, which are well-supported and highly optimized for most DSP/GPU/NPU on mobile devices [6, 20]. To be more specific, the LR image is first processed by M sequential 3×3 convolution with C channels. The final 3×3 convolution projects the feature to the desired dimension, and the output feature is added by the shortcut from the LR input. A PixelShuffle [40] operator is used to generate the final HR image.

The proposed base model is very suitable for mobile scenario with high efficiency and flexibility. The neat topology with low MAC enables super fast inference on mobile devices and the basic operations make it easy to support cross-device deployment. By controlling the values of M and C, the complexity of the model can be conveniently scaled to achieve good trade-off between performance and running speed on different devices.

3.2 Edge-oriented convolution block

Although the plain base model is efficient, its SR performance is less satisfied compared to those complicated models. We thus employ the re-parameterization technique to enrich the representation capability of the base model. Re-parameterization has achieved promising results on several high-level vision tasks [2, 9–11, 53]. However, straightly applying those re-parameterizable blocks designed for high-level vision tasks obtains little improvement in the SR task. We design a more suitable re-parameterization block, namely Edge-oriented Convolution Block (ECB), which can more effectively extract edge and texture information for the SR task. As shown in Fig. 2(b), the ECB consists of four types of carefully designed operators, which are summarized as follows.

Component I: a normal 3×3 convolution. We first employ a normal 3×3 convolution to ensure the base performance. Different from previous re-parameterization blocks in high-level vision tasks, which employ a batch normalization (BN) [19] layer after normal

[1]Most flagship mobile devices use LPDDR4 as off-chip memory, whose bandwidth is far less than GDDR6 and HBM2 used in GPU servers. The memory bandwidth of dual-channel LPDDR4 is about 30GB/s, while the bandwidth of GDDR6 (used in NVIDIA GeForce RTX 2070 / 2080) and HBM2 (used in NVIDIA Tesla P100 / V100) is about 600GB/s and 1TB/s, respectively.

convolution, we do not use BN layer because it hampers the SR performance. The normal convolution is denoted as:

$$F_n = K_n * X + B_n \tag{1}$$

where F_n, X, K_n, B_n represent the output feature, input feature, weights and bias of the normal convolution, respectively.

Component II: expanding-and-squeezing convolution. Wider features can significantly improve the expressiveness and contribute to better performance on SR task [52], we thus design an expanding-and-squeezing convolution as the second component. Specifically, we first employ a $D \times C \times 1 \times 1$ convolution to expand the channel dimension from C to D, then use a $C \times D \times 3 \times 3$ to squeeze the feature back to C channels. Denote by $\{K_e, B_e\}$ and $\{K_s, B_s\}$ as the {weights, bias} of the 1×1 expanding and 3×3 squeezing convolutions, the expanding-and-squeezing feature is extracted as:

$$F_{es} = K_s * (K_e * X + B_e) + B_s \tag{2}$$

In our experiments, we set $D = 2C$ which yields better performance.

Component III: sequential convolution with scaled Sobel filters. Edge information has been proven very helpful for the SR task [32]. Different from [32] which explicitly extracts the 1st-order spatial derivatives using the Sobel filters and employs an additional network branch to process the edge information, we implicitly incorporate extraction of the 1st-order derivatives into the design of our ECB. Since it is difficult for the model to automatically learn sharp edge filters, we alternatively choose to use pre-defined edge filters and learn scaling factor to each of the filter. Specifically, the input feature X is first processed by a $C \times C \times 1 \times 1$ convolution, then the gradient of the intermediate feature is extracted using two scaled sobel filters. Denote by D_x and D_y the horizontal and vertical Sobel filters:

$$D_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad and \quad D_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3}$$

Each channel of the intermediate feature is first processed by the Sobel filters, then scaled by a channel-wise scaling factor. The horizontal and vertical edge information are extracted by:

$$\begin{aligned} F_{Dx} &= (S_{Dx} \cdot D_x) \otimes (K_x * X + B_x) + B_{Dx} \\ F_{Dy} &= (S_{Dy} \cdot D_y) \otimes (K_y * X + B_y) + B_{Dy} \end{aligned} \tag{4}$$

where $\{K_x, B_x\}$ and $\{K_y, B_y\}$ are the {weights, bias} of 1×1 convolution for horizontal and vertical branch, $\{S_{Dx}, B_{Dx}\}$ and $\{S_{Dy}, B_{Dy}\}$ are the scaling parameters and bias with the shape of $C \times 1 \times 1 \times 1$, $\otimes$ and $*$ represent depth-wise convolution (DWConv) and normal convolution, $\cdot$ indicates channel-wise broadcasting multiplication, $\{(S_{Dx} \cdot D_x), (S_{Dy} \cdot D_y)\}$ are in the shape of $C \times 1 \times 3 \times 3$. The combined edge information extracted by the scaled Sobel filters is:

$$F_{sob} = F_{Dx} + F_{Dy} \tag{5}$$

Component IV: sequential convolution with scaled Laplacian filters. In addition to the 1st-order derivatives, we also extract the 2nd-order spatial derivative using a Laplacian filter which is more stable and robust to noise for edge information extraction [42]. Similarly, the input feature X is first processed using a $C \times C \times 1 \times 1$ convolution, then the 2nd-order spatial derivative is extracted using a Laplician filter as follow:

$$D_{lap} = \begin{bmatrix} 0 & +1 & 0 \\ +1 & -4 & +1 \\ 0 & +1 & 0 \end{bmatrix} \tag{6}$$

The scaled 2nd-order edge information is extracted by:

$$F_{lap} = (S_{lap} \cdot D_{lap}) \otimes (K_l * X + B_l) + B_{lap} \tag{7}$$

where $\{K_l, B_l\}$ are the {weights, bias} of the 1×1 convolution, $\{S_{lap}, B_{lap}\}$ are scaling factors and bias of DWConv, respectively.

The output of the ECB in the combination of the four components:

$$F = F_n + F_{es} + F_{sob} + F_{lap} \tag{8}$$

The combined feature map is then feed into a non-linear activation layer. PReLU is employed in our experiments.

3.3 Re-parameterization for efficient inference

We now describe how to re-parameterize the ECB into a single 3×3 convolution for efficient inference. According to [10], the 1×1 expanding and 3×3 squeezing convolutions in component II can be merged into one single normal convolution with parameters $\{K_{es}, B_{es}\}$,

$$\begin{aligned} K_{es} &= perm(K_e) * K_s, \\ B_{es} &= K_s * rep(B_e) + B_s \end{aligned} \tag{9}$$

where $perm$ denotes the permute operation which exchanges the 1st and 2nd dimensions of a tensor, thus the shape of $perm(K_e)$ is $C \times D \times 1 \times 1$. rep is the spatial broadcasting operation, which replicates the bias $B_e \in \mathbb{R}^{1 \times D \times 1 \times 1}$ into $rep(B_e) \in \mathbb{R}^{1 \times D \times 3 \times 3}$.

Both components III and IV employ a sequence of 1×1 convolution and 3×3 DWConv. The DWConv can be regarded as a normal convolution with a special sparsity constraint on the channel dimension. Denote by $\{K_{Dx}, K_{Dy}, K_{lap}\} \in \mathbb{R}^{C \times C \times 3 \times 3}$ the weights of the normal convolutions which are equal to the DWConv used to extract $\{F_{Dx}, F_{Dy}, F_{lap}\}$, respectively, we have:

$$\begin{aligned} &K_{Dx}[i,i,:,:] = (S_{Dx} \cdot D_x)[i,1,:,:] \quad and \quad K_{Dx}[i,j,:,:] = 0, i \neq j \\ &K_{Dy}[i,i,:,:] = (S_{Dy} \cdot D_y)[i,1,:,:], \quad and \quad K_{Dy}[i,j,:,:] = 0, i \neq j \\ &K_{lap}[i,i,:,:] = (S_{lap} \cdot D_{lap})[i,1,:,:] \quad and \quad K_{lap}[i,j,:,:] = 0, i \neq j \end{aligned} \tag{10}$$

where $i, j = 1, 2, ..., C$. The final weights and bias after re-parameterization are as follows:

$$\begin{aligned} K_{rep} &= K_n + K_{es} + perm(K_x) * K_{Dx} \\ &\quad + perm(K_y) * K_{Dy} + perm(K_l) * K_{lap} \\ B_{rep} &= B_n + B_{es} + (K_{Dx} * rep(B_x) + B_{Dx}) \\ &\quad + (K_{Dy} * rep(B_y) + B_{Dy}) + (K_{lap} * rep(B_l) + B_{lap}) \end{aligned} \tag{11}$$

The output feature can be obtained using one single normal convolution in the inference stage as follow:

$$F = K_{rep} * X + B_{rep} \tag{12}$$

Scale	Model	#Params(K)	#FLOPs(G)	#Acts(M)	#Conv	Set5	Set14	B100	U100	DIV2K
× 2	Bicubic	—	—	—	—	33.68/0.9307	30.24/0.8693	29.56/0.8439	26.88/0.8408	32.45/0.9043
	SRCNN [12]	24.00	52.70	89.39	3	36.66/0.9542	32.42/0.9063	31.36/0.8879	29.50/0.8946	34.61/0.9334
	ESPCN [40]	21.18	4.55	23.04	3	36.83/0.9564	32.40/0.9096	31.29/0.8917	29.48/0.8975	34.63/0.9342
	ECBSR-M4C8	2.80	0.64	10.14	6	36.93/0.9577	32.51/0.9107	31.44/0.8932	29.68/0.9014	34.80/0.9356
	FSRCNN [13]	12.46	6.00	40.53	8	36.98/0.9556	32.62/0.9087	31.50/0.8904	29.85/0.9009	34.74/0.9340
	MOREMNAS-C [8]	25.00	5.50	269.11	49	37.06/0.9561	32.75/0.9094	31.50/0.8904	29.92/0.9023	34.87/0.9356
	ECBSR-M4C16	10.20	2.34	19.35	6	37.33/0.9593	32.81/0.9129	31.66/0.8961	30.31/0.9091	35.15/0.9382
	TPSR-NoGAN [27]	60.00	14.00	50.69	14	37.38/0.9583	33.00/0.9123	31.75/0.8942	30.61/0.9119	—
	IMDN-RTC [16]	19.70	4.57	65.20	28	37.51/0.9600	32.93/0.9144	31.79/0.8980	30.67/0.9140	35.34/0.9398
	ECBSR-M10C16	24.30	5.60	41.47	12	37.55/0.9602	32.98/0.9144	31.85/0.8985	30.78/0.9149	35.38/0.9402
	LapSRN [24]	813.00	29.90	223.03	14	37.52/0.9590	33.08/0.9130	31.80/0.8950	30.41/0.9100	35.31/0.9400
	FALSR-B [7]	326.00	74.70	372.33	49	37.61/0.9585	33.29/0.9143	31.97/0.8967	31.28/0.9191	35.58/0.9408
	FALSR-C [7]	408.00	93.70	379.70	34	37.66/0.9586	33.26/0.9140	31.96/0.8965	31.24/0.9187	35.57/0.9407
	EDSR-R5C32 [30]	130.80	30.31	111.51	13	37.61/0.9605	33.06/0.9144	31.87/0.8970	30.90/0.9166	35.45/0.9407
	ECBSR-M10C32	94.70	21.81	82.02	12	37.76/0.9609	33.26/0.9146	32.04/0.8986	31.25/0.9190	35.68/0.9421
	VDSR [21]	665.00	612.60	1121.59	20	37.53/0.9587	33.05/0.9127	31.90/0.8960	30.77/0.9141	35.43/0.9410
	EDSR-R16C64 [30]	1334.90	307.89	546.51	37	37.99/0.9604	33.57/0.9175	32.16/0.8994	31.98/0.9272	35.85/0.9436
	MOREMNAS-B [8]	1118.00	256.90	987.96	79	37.58/0.9584	33.22/0.9135	31.91/0.8959	31.14/0.9175	35.46/0.9402
	CARN-M [1]	412.00	91.20	649.73	42	37.53/0.9583	33.26/0.9141	31.92/0.8960	31.23/0.9193	35.62/0.9420
	IMDN [16]	660.30	152.04	406.43	34	37.99/0.9619	33.39/0.9179	32.14/0.9022	32.03/0.9279	35.87/0.9436
	ECBSR-M16C64	596.00	137.31	251.60	18	37.90/0.9615	33.34/0.9178	32.10/0.9018	31.71/0.9250	35.79/0.9430
× 4	Bicubic	—	—	—	—	28.43/0.8113	26.00/0.7025	25.96/0.6682	23.14/0.6577	28.10/0.7745
	SRCNN [12]	57.00	52.7	89.39	3	30.48/0.8628	27.49/0.7503	26.90/0.7101	24.52/0.7221	29.25/0.8090
	ESPCN [40]	24.90	1.44	6.45	3	30.52/0.8697	27.42/0.7606	26.87/0.7216	24.39/0.7241	29.32/0.8120
	ECBSR-M4C8	3.70	0.21	3.23	6	30.52/0.8698	27.43/0.7608	26.89/0.7220	24.41/0.7263	29.35/0.8133
	FSRCNN [13]	12.00	5.00	10.81	8	30.70/0.8657	27.59/0.7535	26.96/0.7128	24.60/0.7258	29.36/0.8110
	ECBSR-M4C16	11.90	0.69	5.53	6	31.04/0.8805	27.78/0.7693	27.09/0.7283	24.79/0.7422	29.62/0.8197
	TPSR-NoGAN [27]	61.00	3.60	13.13	15	31.10/0.8779	27.95/0.7663	27.15/0.7214	24.97/0.7456	29.77/0.8200
	IMDN-RTC [16]	21.00	1.22	16.99	28	31.22/0.8844	27.92/0.7730	27.18/0.7314	24.98/0.7504	29.76/0.8230
	ECBSR-M10C16	26.00	1.50	11.06	12	31.37/0.8866	27.99/0.7740	27.22/0.7329	25.08/0.7540	29.80/0.8241
	LapSRN [24]	813.00	149.40	264.04	27	31.54/0.8850	28.19/0.7720	27.32/0.7280	25.21/0.7560	29.88/0.8250
	EDSR-R5C32 [30]	241.80	14.15	50.69	13	31.46/0.8880	28.07/0.7760	27.27/0.7340	25.21/0.7579	29.87/0.8256
	ECBSR-M10C32	98.10	5.65	21.20	12	31.66/0.8911	28.15/0.7776	27.34/0.7363	25.41/0.7653	29.98/0.8281
	VDSR [21]	665.00	612.60	1121.59	20	31.35/0.8838	28.02/0.7678	27.29/0.7252	25.18/0.7525	29.82/0.8240
	EDSR-R16C64 [30]	1778.00	102.85	181.56	37	32.09/0.8938	28.58/0.7813	27.57/0.7357	26.04/0.7849	30.21/0.8336
	CARN-M [1]	412.00	46.10	222.11	43	31.92/0.8903	28.42/0.7762	27.44/0.7304	25.62/0.7694	30.10/0.8311
	IMDN [16]	667.40	38.41	102.30	34	32.03/0.8966	28.42/0.7842	27.48/0.7409	25.96/0.7843	30.22/0.8336
	ECBSR-M16C64	602.90	34.73	63.59	18	31.92/0.8946	28.34/0.7817	27.48/0.7393	25.81/0.7773	30.15/0.8315

Table 1: Performance comparison of different SR models on five benchmarks. PSNR/SSIM on Y channel are reported on each dataset. #Params, #FLOPs, #Acts and #Conv represent the total number of network parameters, floating-point operations, activation and convolution layers, respectively. The #FLOPs and #Acts are measured under the setting of generating an SR image of 1280×720 resolution on both scales. The results of our model and best candidate are in blue and red color.

4 EXPERIMENTS

In this section, we first conduct extensive experiments to validate the superior performance of our ECBSR model on five SR benchmark datasets and its high efficiency on two typical hardware. We then perform comprehensive ablation studies to valid the design of our proposed ECB.

4.1 Datasets and implementation details

We train our models on DIV2K datasets [45] with 800 training images. The validation set of DIV2K together with several standard benchmark datasets, including Set5 [4], Set14 [54], BSD100 [34] and Urban100 [15], are used for performance evaluation. We use PSNR and SSIM [50] as the evaluation metrics, and calculate them on the Y channel after converting RGB to YCbCr format.

We randomly crop 32 patches of size 64×64 from the LR images as input for each training mini-batch. Data augmentation is performed on the training set, such as random rotations of 90°, 180° and 270°, and horizontal flips. All models are trained by the ADAM [23] optimizer with standard L_1 loss for 700 epoches. The learning rate is set to a constant 5×10^{-4}. The model training is conducted by using Pytorch [37] toolbox on a NVIDIA Tesla P100 GPU.

4.2 Benchmark results

We compare the proposed ECBSR with representative SR models, including SRCNN [12], FSRCNN [13], ESPCN [40], VDSR [21], LapSRN [24], CARN-M [1], MoreMNAS-{B,C} [8], FALSR-{B,C} [7], TPSR-NoGAN [27] , EDSR [30] and IMDN [16], on ×2 and ×4 upscaling tasks. Since the default version of EDSR and IMDN are designed for GPU server and have very deep or complicated topology, we slim EDSR into two lightweight versions, EDSR-R5C32 and EDSR-R16C64, and also report the IMDN-RTC (a lightweight version of IMDN) for comparison. EDSR-R5C32 contains 5 residual

blocks with 32 channels for each convolution layer. In order to make fair comparison with previous models, we scale our ECBSR to five different levels of complexity: ECBSR-M4C8, ECBSR-M4C16, ECBSR-M10C16, ECBSR-M10C32, and ECBSR-M16C64.

The performance comparison of different SR models on 5 benchmark datasets are summarized in Table. 1. In addition to PSNR/SSIM indexes, we also list the number of parameters, FLOPs, activation and convolution layers for more comprehensive comparison. The number of FLOPs and activation are calculated under the setting of upscaling image to 1280×720 resolution on both $\times 2$ and $\times 4$ tasks. It has been recently shown that the number of activation is a better metric for evaluating the model efficiency than number of parameters and FLOPs [38, 55].

Several interesting observations can be made from Table. 1. As the smallest version of our model, ECBSR-M4C8 outperforms both SRCNN [12] and ESPCN [40] by a large margin on all five benchmarks, while using about $12\times/10\times$ fewer parameters, $92\times/9\times$ fewer FLOPs, $9\times/2\times$ fewer activations, respectively. Using only a little more computation and memory, ECBSR-M4C8 can achieve much better performance than bicubic upsampling. Similarly, ECBSR-M4C16, ECBSR-M10C16 and ECBSR-M10C32 show obvious advantages over their competitors on both performance and model complexity in most cases. It is worth mentioning that MOREMNAS-C, FALSR-B and FALSR-C are obtained using neural architecture search. Our ECBSR models are either comparable or better than them while having much smaller model size, computation and memory consumption. We also scale up our ECBSR to M16C64 in order to compare with some more complicated SR models. Using similar number of layers and much less parameters, FLOPs and activations, ECBSR-M16C64 outperforms VDSR by a large margin on all five datasets. Our ECBSR-M16C64 can obtain comparable performance to EDSR-R16C64, CARN-M and IMDN but it is much smaller and lightweight. Although further improving the number of block and feature channel of ECBSR can brings better performance, the computation and memory cost will be too heavy for mobile devices as will be discussed in the next section.

4.3 Hardware running speed

Since model size, parameters, FLOPs and activations cannot faithfully reflect the real running speed of SR models, we further evaluate their real running speed on two mobile devices. We select two representative *flagship* mobile SoC for evaluation, the GPU of Dimensity 1000+ and the DSP of SnapDragon 865. Since SDK also plays an important role in inference speed, we run all models with the same SDK under the same setting. We use the AI benchmark app [18] for model execution. The TFLITE GPU and Hexagon NN are selected as the delegates of inference engine. Moreover, all models are quantized and executed in 8-bit arithmetic. The running speed of upscaling images to 1080p resolution on both $\times 2$ and $\times 4$ tasks are reported in Table 2.

As can be seen, most of the compared SR models are far from real-time on both mobile devices. Our ECBSR-M4C8 and ECBSR-M4C16 are super efficient, achieving real-time speed on both devices. ECBSR-M10C16 and ECBSR-M10C32 can also achieve nearly real-time performance in most cases, while ECBSR-M16C64 is too heavy for both mobile devices. Regarding the compared models, only ESPCN and FSRCNN can achieve real-time performance in some cases ($\times 4$ tasks) because of their very neat topology. Although using neat topology, SRCNN is much slower owing to its pre-upsampling strategy, which substantially increases memory and computation consumption. Even using a lightweight version, EDSR-R5C32 and IMDN-RTC are unable to reach nearly real-time speed on both devices because their dense connections and multi-branch topology introduce too many MACs and decrease execution parallelism. The other models suffer from the same problem and are even slower.

Model	Snapdragon 865 DSP (/second)		Dimensity 1000+ GPU (/second)	
	$\times 2$	$\times 4$	$\times 2$	$\times 4$
SRCNN [12]	1.591	1.583	0.890	0.896
ESPCN [40]	0.072	0.026	0.080	0.032
ECBSR-M4C8	0.032	0.010	0.022	0.011
FSRCNN [13]	0.114	0.032	0.076	0.028
ECBSR-M4C16	0.033	0.011	0.046	0.017
IMDN-RTC [16]	1.101	0.318	1.076	0.287
ECBSR-M10C16	0.060	0.017	0.089	0.029
LapSRN [24]	4.395	5.378	1.624	5.801
EDSR-R5C32 [30]	0.150	0.101	0.653	0.567
ECBSR-M10C32	0.062	0.017	0.203	0.063
VDSR [21]	8.946	9.036	3.379	3.365
CARN-M [1]	0.884	0.170	1.195	0.362
EDSR-R16C64 [30]	1.447	0.527	2.372	1.639
IMDN [16]	10.610	2.782	12.792	2.676
ECBSR-M16C64	0.513	0.071	0.786	0.209

Table 2: Comparison of hardware running speed of SR models on upscaling image to 1920×1080 using two flagship mobile devices. Real-time performance (> 30 fps) and nearly real-time performance (15 fps $\leq$ x $\leq$ 30 fps) are in red and blue.

4.4 Qualitative results

In this section, we further qualitatively compare the SR quality of different models. Since we focus on real-time SR on mobile devices, we only compare SR models that can reach real-time or nearly real-time speed, including bicubic upsampling, FSRCNN, ESPCN, ECBSR-M4C8, ECBSR-M4C16, ECBSR-M10C16 and ECBSR-M10C32. The $\times 2$ SR results on two typical example images from Urban100 are shown in Fig. 3. One can see that ESPCN, FSRCNN and ECBSR-M4C8 have only slightly better visual quality than bicubic upsampling, and all of them fail to recover sufficiently distinct edges on image033 because of their very shallow topology. By improving the depth and width of the topology, ECBSR-M4C16, ECBSR-M10C16 and ECBSR-M10C32 achieve consistently better visual quality, with sharper edges and clearer textures. For the results of img004, ESPCN and FSRCNN introduce slightly undesired texture on the smooth area and waved ceiling pattern around lattices. The proposed ECBSR has better perceptual results than ESPCN and FSRCNN. ECBSR-M10C32 performs the best among all models, and faithfully restores the structure and edge details of the lattices.

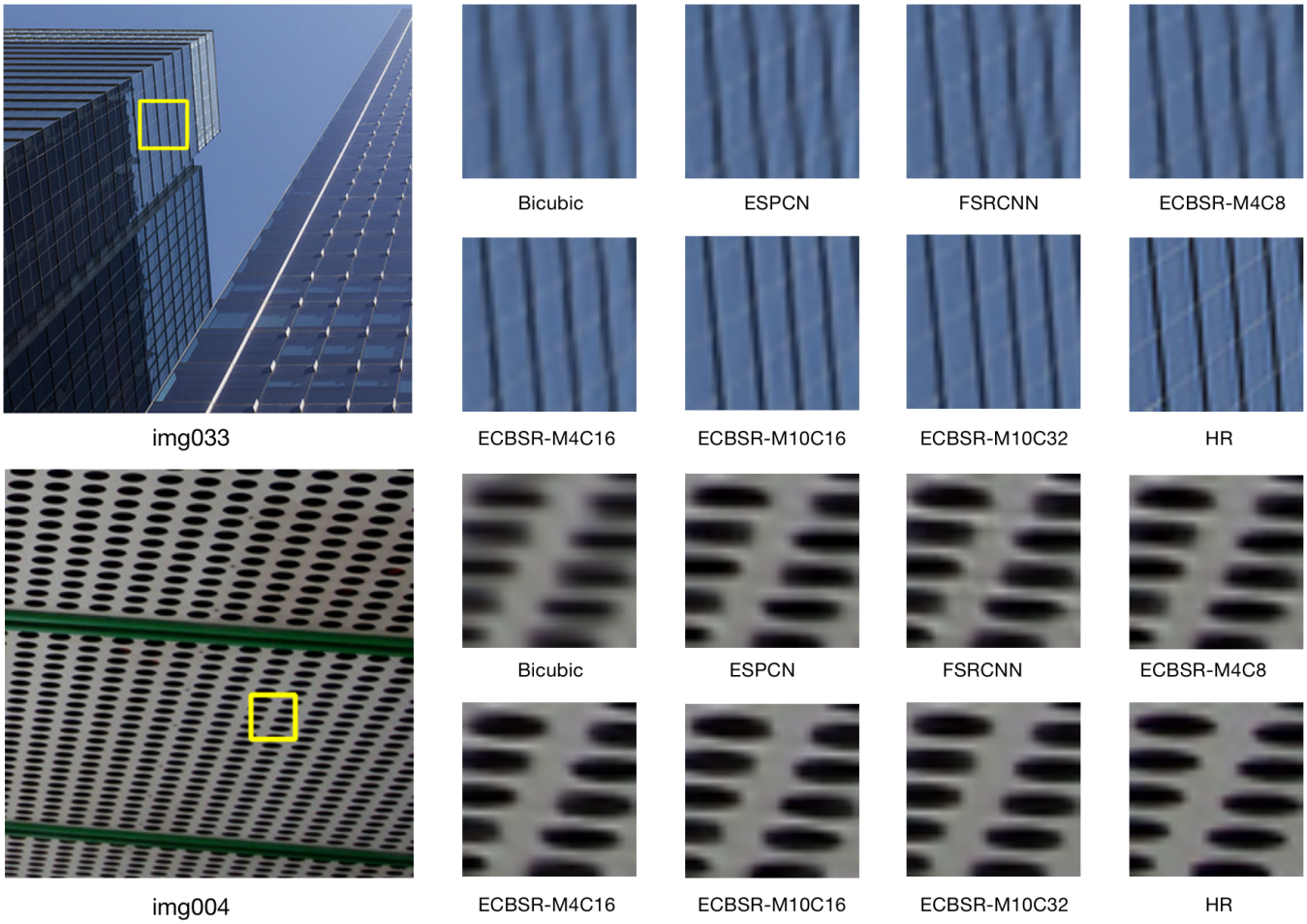


Figure 3: Qualitative comparison of real-time and nearly real-time SR models on Urban100 for $\times$ 2 upscaling task.

4.5 Ablation studies

We finally conduct a series of ablation studies on the design of the proposed ECB. Specifically, we first ablate some branches of ECB and observe the change in performance, then compare ECB with some other re-parameterization blocks proposed in high-level vision tasks, including the AC block [9], the RepVGG Block [11] and the DB block [10]. Triple duplicate 3×3 convolution block is also compared as a baseline reference. Considering that BN hurts the SR performance, we discard the BN layers for all competitors. The compared re-parameterization blocks are visualizaed in Fig. 4 and the PSNR/SSIM indexes on five benchmarks are reported in Table 3. All models are trained from scratch using the same setting. PSNR/SSIM are evaluated on the $\times 2$ upscaling task.

From the results in Table 3, one can observe that employing any of the three components can boost the performance of the baseline model, and removing any component in ECB degrades its performance. This implies that all components in ECB are helpful for the SR task and those components are complementary to each other. As for the other re-parameterization blocks, triple duplicate 3×3 convolution block, AC block and RepVGG block obtain only marginal improvement compared to the baseline. The DB block achieves slightly better performance, leading to less than 0.05dB improvement on the PSNR index. In comparison, our ECB can consistently improve PSNR by about 0.1dB on all datasets. It is worth mentioning that all the compared blocks are merged into one single 3×3 normal convolution and have the same computation and memory cost at inference stage. The advantage of ECB validates the effectiveness of our edge-oriented design for the SR task.

We also conduct some experiments by applying our ECB to some existing SR topology to further validate the effectiveness of ECB. Specifically, we replace the 3×3 normal convolutions in FSRCNN and ESPCN with our ECB and name the enhanced version as FSRCNN+ and ESPCN+, respectively. For fair comparison, we re-implemented all models and trained them under the same setting. The PSNR/SSIM comparison of different models on five datasets are reported in Table. 4. As can be seen, the enhanced models again obtains $\geq$0.1dB consistent improvement on the PSNR index on most datasets. This indicates that our ECB is a general and drop-in replacement module for improving SR performance without introducing additional inference cost.

Block	3×3 conv	Expand-and-Squeeze	1×1—Sobel	1×1—Laplacian	Set5	Set14	B100	Urban100	DIV2K
ECB	1	✓	✓	✓	37.33/0.9593	32.81/0.9129	31.66/0.8961	30.31/0.9091	35.15/0.9382
ECB	1	✓	✓		37.29/0.9592	32.75/0.9129	31.64/0.8960	30.26/0.9090	35.11/0.9381
ECB	1	✓		✓	37.29/0.9592	32.77/0.9129	31.64/0.8959	30.23/0.9087	35.09/0.9380
ECB	1		✓	✓	37.28/0.9592	32.77/0.9130	31.64/0.8960	30.23/0.9086	35.08/0.9380
ECB	1	✓			37.26/0.9591	32.75/0.9129	31.63/0.8959	30.24/0.9085	35.09/0.9381
ECB	1		✓		37.27/0.9591	32.75/0.9128	31.62/0.8957	30.21/0.9082	35.07/0.9379
ECB	1			✓	37.28/0.9592	32.76/0.9127	31.64/0.8959	30.23/0.9085	35.09/0.9380
Baseline	1				37.24/0.9572	32.73/0.9101	31.60/0.8956	30.15/0.9075	35.05/0.9377
Triple Duplicate	3				37.26/0.9591	32.74/0.9127	31.62/0.8958	30.21/0.9085	35.07/0.9379
AC Block [9]	1				37.24/0.9590	32.75/0.9127	31.63/0.8958	30.22/0.9084	35.08/0.9379
RepVGG Block [11]	1				37.26/0.9591	32.75/0.9128	31.62/0.8958	30.17/0.9079	35.07/0.9379
DB Block [10]	1				37.29/0.9592	32.74/0.9127	31.63/0.8957	30.23/0.9086	35.09/0.9380

Table 3: Ablation studies on the design of ECB based on the M4C16 topology on five benchmarks. All the compared blocks can be merged into one single 3×3 convolution in the inference stage.

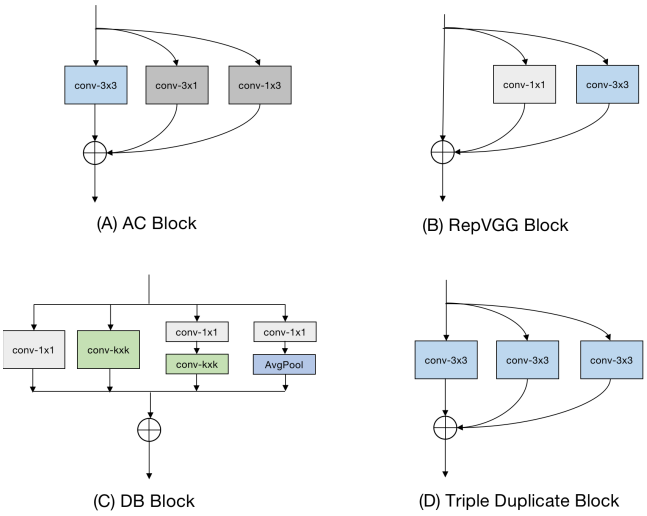


Figure 4: Illustration of the compared re-perameterization blocks.

Model	Set5	Set14	B100	Urban100	DIV2K
FSRCNN	37.12/ 0.9586	32.64/ 0.9122	31.53/ 0.8948	30.12/ 0.9067	35.02/ 0.9374
FSRCNN+	37.26/ 0.9592	32.77/ 0.9129	31.64/ 0.8960	30.28/ 0.9087	35.12/ 0.9386
ESPCN	36.83/ 0.9564	32.40/ 0.9096	31.29/ 0.8917	29.48/ 0.8975	34.63/ 0.9342
ESPCN+	36.92/ 0.9573	32.51/ 0.9106	31.38/ 0.8928	29.59/ 0.8989	34.73/ 0.9348

Table 4: Performance comparison of applying our ECB to other SR models.

5 CONCLUSION

In this paper, we proposed an Edge-oriented Convolution Block (ECB) for efficient and light-weight SR design towards mobile devices. Based on the proposed ECB, we further designed an ECBSR model, aiming at balancing hardware efficiency and PSNR/SSIM indexes. Extensive experiments across five benchmarks were conducted to validate the efficiency and effectiveness of ECBSR over state-of-the-art lightweight SR models. Our ECBSR achieves real-time SR on two flapship mobile SoC and maintains high visual quality. In the future, we will explore more efficient and effective re-parameterization blocks and network topologies using the network architecture search technique.

ACKNOWLEDGMENTS

This work is supported by the Hong Kong RGC RIF grant (R5001-18).

REFERENCES

[1] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. 2018. Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 252–268.
[2] Sanjeev Arora, Nadav Cohen, and Elad Hazan. 2018. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*. PMLR, 244–253.
[3] Roberto H Bamberger and Mark JT Smith. 1992. A filter bank for the directional decomposition of images: Theory and design. *IEEE transactions on signal processing* 40, 4 (1992), 882–893.
[4] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. 2012. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. (2012).
[5] Jose Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4778–4787.
[6] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759* (2014).
[7] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, and Qingyuan Li. 2019. Fast, accurate and lightweight super-resolution with neural architecture search. *arXiv preprint arXiv:1901.07261* (2019).

[8] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. 2020. Multi-objective reinforced evolution in mobile neural architecture search. In *European Conference on Computer Vision*. Springer, 99–113.

[9] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. 2019. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1911–1920.

[10] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. 2021. Diverse Branch Block: Building a Convolution as an Inception-like Unit. *arXiv preprint arXiv:2103.13425* (2021).

[11] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. 2021. RepVGG: Making VGG-style ConvNets Great Again. *arXiv preprint arXiv:2101.03697* (2021).

[12] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2015. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence* 38, 2 (2015), 295–307.

[13] Chao Dong, Chen Change Loy, and Xiaoou Tang. 2016. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*. Springer, 391–407.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[15] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. 2015. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5197–5206.

[16] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. 2019. Lightweight image super-resolution with information multi-distillation network. In *Proceedings of the 27th ACM International Conference on Multimedia*. 2024–2032.

[17] Zheng Hui, Xiumei Wang, and Xinbo Gao. 2018. Fast and accurate single image super-resolution via information distillation network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 723–731.

[18] Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. 2019. Ai benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 3617–3635.

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

[20] Xiaotang Jiang, Huan Wang, Yiliu Chen, Ziqi Wu, Lichuan Wang, Bin Zou, Yafeng Yang, Zongyang Cui, Yu Cai, Tianhang Yu, et al. 2020. MNN: A universal and efficient inference engine. *arXiv preprint arXiv:2002.12418* (2020).

[21] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1646–1654.

[22] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. 2016. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1637–1645.

[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[24] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. 2017. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 624–632.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[26] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4681–4690.

[27] Royson Lee, Łukasz Dudziak, Mohamed Abdelfattah, Stylianos I Venieris, Hyeji Kim, Hongkai Wen, and Nicholas D Lane. 2020. Journey Towards Tiny Perceptual Super-Resolution. In *European Conference on Computer Vision*. Springer, 85–102.

[28] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. 2020. Dhp: Differentiable meta pruning via hypernetworks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*. Springer, 608–624.

[29] Yudong Liang, Jinjun Wang, Sanping Zhou, Yihong Gong, and Nanning Zheng. 2016. Incorporating image priors with deep convolutional neural networks for image super-resolution. *Neurocomputing* 194 (2016), 340–347.

[30] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. 2017. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 136–144.

[31] Jie Liu, Jie Tang, and Gangshan Wu. 2020. Residual feature distillation network for lightweight image super-resolution. *arXiv preprint arXiv:2009.11551* (2020).

[32] Cheng Ma, Yongming Rao, Yean Cheng, Ce Chen, Jiwen Lu, and Jie Zhou. 2020. Structure-preserving super resolution with gradient guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7769–7778.

[33] Yinglan Ma, Hongyu Xiong, Zhe Hu, and Lizhuang Ma. 2019. Efficient super resolution using binarized neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.

[34] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, Vol. 2. IEEE, 416–423.

[35] Ying Nie, Kai Han, Zhenhua Liu, An Xiao, Yiping Deng, Chunjing Xu, and Yunhe Wang. 2021. GhostSR: Learning Ghost Features for Efficient Image Super-Resolution. *arXiv preprint arXiv:2101.08525* (2021).

[36] Sylvain Paris, Samuel W Hasinoff, and Jan Kautz. 2011. Local Laplacian filters: Edge-aware image processing with a Laplacian pyramid. *ACM Trans. Graph.* 30, 4 (2011), 68.

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

[38] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10428–10436.

[39] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. 2018. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6626–6634.

[40] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1874–1883.

[41] Dehua Song, Yunhe Wang, Hanting Chen, Chang Xu, Chunjing Xu, and DaCheng Tao. 2020. Addersr: Towards energy efficient image super-resolution. *arXiv preprint arXiv:2009.08891* (2020).

[42] Jian Sun, Zongben Xu, and Heung-Yeung Shum. 2008. Image super-resolution using gradient profile prior. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

[43] Ying Tai, Jian Yang, and Xiaoming Liu. 2017. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3147–3155.

[44] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. 2017. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*. 4472–4480.

[45] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. 2017. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 114–125.

[46] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. 2017. Image super-resolution using dense skip connections. In *Proceedings of the IEEE international conference on computer vision*. 4799–4807.

[47] Chaofeng Wang, Zheng Li, and Jun Shi. 2019. Lightweight image super-resolution with adaptive weighted learning network. *arXiv preprint arXiv:1904.02358* (2019).

[48] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. 2019. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.

[49] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. 2018. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 0–0.

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

[51] Jingwei Xin, Nannan Wang, Xinrui Jiang, Jie Li, Heng Huang, and Xinbo Gao. 2020. Binarized neural network for single image super resolution. In *European Conference on Computer Vision*. Springer, 91–107.

[52] Jiahui Yu, Yuchen Fan, and Thomas Huang. 2020. Wide activation for efficient image and video super-resolution. In *30th British Machine Vision Conference, BMVC 2019*.

[53] Sergey Zagoruyko and Nikos Komodakis. 2017. Diracnets: Training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388* (2017).

[54] Roman Zeyde, Michael Elad, and Matan Protter. 2010. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*. Springer, 711–730.

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

[56] Kai Zhang, Shuhang Gu, Radu Timofte, Zheng Hui, Xiumei Wang, Xinbo Gao, Dongliang Xiong, Shuai Liu, Ruipeng Gang, Nan Nan, et al. 2019. Aim 2019

challenge on constrained super-resolution: Methods and results. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE, 3565–3574.

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

[58] Fuqiang Zhou, Xiaojie Li, and Zuoxin Li. 2018. High-frequency details enhancing DenseNet for super-resolution. *Neurocomputing* 290 (2018), 34–42.