2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) | 979-8-3503-0129-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/CVPR52729.2023.01355

# Equivalent Transformation and Dual Stream Network Construction for Mobile Image Super-Resolution

Jiahao Chao[1], Zhou Zhou[1], Hongfan Gao[1], Jiali Gong[1], Zhengfeng Yang[1]*, Zhenbing Zeng[2], Lydia Dehbi[1]
[1]East China Normal University;
[2]Chengdu Institute of Computer Applications of Chinese Academy of Sciences
{jhchao502, zhouzhou, hfgao, gongjl}@stu.ecnu.edu.cn, zfyang@sei.ecnu.edu.cn,
zbzeng@shu.edu.cn, dehbilydia@sei.ecnu.edu.cn

## Abstract

*In recent years, there has been an increasing demand for real-time super-resolution networks on mobile devices. To address this issue, many lightweight super-resolution models have been proposed. However, these models still contain time-consuming components that increase inference latency, limiting their real-world applications on mobile devices. In this paper, we propose a novel model for single-image super-resolution based on Equivalent Transformation and Dual Stream network construction (ETDS). ET method is proposed to transform time-consuming operators into time-friendly operations, such as convolution and ReLU, on mobile devices. Then, a dual stream network is designed to alleviate redundant parameters resulting from the use of ET and enhance the feature extraction ability. Taking full advantage of the advance of ET and the dual stream network structure, we develop the efficient SR model ETDS for mobile devices. The experimental results demonstrate that our ETDS achieves superior inference speed and reconstruction quality compared to previous lightweight SR methods on mobile devices. The code is available at https://github.com/ECNUSR/ETDS.*

## 1. Introduction

Image super-resolution (SR) aims to reconstruct high-resolution images (HR) from low-resolution images (LR). Over the years, numerous deep-learning methods have been proposed [3, 6, 17, 18, 33, 35, 36] with good fidelity and perceptual quality. However, these methods are not efficient and lightweight when it comes to mobile platforms where SR application becomes increasingly ubiquitous. Thus, it is essential to devise an approach that takes into account the restrictions of mobile platforms.

Generally, mobile platforms have limitations such as a restricted amount of RAM, lower memory bandwidth, lower computational speed and insufficient support for many common deep learning layers and operators. To take the particularities into consideration, the recently proposed SR models [27, 34] designed for mobile devices adopt a neat topology [34] as the base model to ensure low inference latency. ABPN [8] further boosted efficiency by employing the *repeat* operator instead of the time-consuming nearest neighbor interpolation. Nevertheless, in-depth investigation reveals that some time-consuming components in current mobile SR models, such as the global residual connection and clip operator, are indispensable for overall reconstruction quality. Therefore, to accelerate the inference on mobile devices and achieve a competitive reconstruction quality and inference latency, it is necessary to seek time-friendly surrogates for these time-consuming operators.

To this end, we propose Equivalent Transformation (ET), a method that speeds up the model by substituting time-consuming operators with time-friendly ones without im-

*Corresponding author.

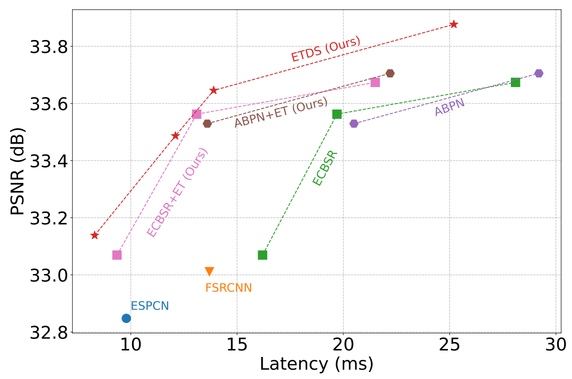

Figure 1. Comparisons of PSNR performance and the inference latency of different models. The inference latency is tested on Dimensity 8100 SoC, NNAPI driver, INT8 precision and upsampling from 360 × 640 to 1080 × 1920. PSNR indexes are evaluated on Set5 [2].

pairing reconstruction quality. As shown in Fig. 1, the proposed ET can be directly applied to existing models (*e.g.*, ECBSR [34] and ABPN [8]) and reduce inference latency without retraining. However, ET introduces some redundant and unlearnable parameters. To fully utilize these parameters, we design the dual stream network that makes the redundant parameters partially learnable, to boost the feature extraction ability. Finally, we propose a mobile image SR model named ETDS that employs the dual stream network in the training stage and transforms it into an equivalent plain network by ET in the inference stage. As shown in Fig. 1, our ETDS not only achieves high reconstruction quality but also maintains low inference speed.

In summary, the main contributions of this paper are as follows:

1) We propose ET, a method that can transform time-consuming operators and speed up the inference without impairing reconstruction quality. It can be applied to existing models to accelerate the inference.
2) We design a dual stream network to alleviate the redundancy yielded from ET by making redundant parameters partially learnable.
3) We propose an efficient and lightweight network named ETDS for real-time SR on mobile devices based on ET and dual stream networks. Experiments demonstrate that state-of-the-art models equipped with ET have at most 80% improvement in inference latency and ETDS achieves 34% inference latency improvement and 0.42dB PSNR performance improvement.

## 2. Related Work

Due to the rapid development of convolutional neural networks (CNN), CNN-based methods [6, 18, 24, 35, 36] have become mainstream methods for SR tasks. SRCNN [6] pioneered the application of convolutional neural networks on the SR task, surpassing the performance of traditional methods. In EDSR [18], a very deep network was utilized and the batch normalization layers in the residual block were removed. SwinIR [17] first attempted to apply the Swin-Transformer [20] on the SR task, showing the potential of Transformer-based networks [26]. To enlarge the receptive field, the hybrid attention block (HAB) was proposed by HAT [3], which achieved state-of-the-art performance. However, these models have high requirements for memory and computational resources which are not easily attainable in real-world applications.

To realize lightweight SR for GPU servers, many approaches tried to reduce the number of parameters and FLOPs. CARN [1] attempted to apply group convolutions. IMDN [10] employed a progressive refinement module to improve the information extraction ability and reduced the number of layers and channels. For further improvement in the efficiency of feature extraction, in RFDN [19], $1 \times 1$ convolutions were utilized to replace the $3 \times 3$ convolutions on the split channel of the IMDB [10] module. RLFN [15] replaced the progressive refinement module with a simpler residual module, which further improved the running speed and achieved great performance. MemSR [30] optimized the network in terms of memory by removing the residual structure in the model and proposed a novel knowledge distillation method to improve the performance.

Due to the particularity of the mobile platforms, most models for GPU servers are not directly applicable to mobile devices whose applications require to be carried out in a timely fashion with restricted computational resources. To solve this problem, current methods mainly optimized the network from three aspects, *i.e.*, neat network topology, computation reduction and operator substitution. ECBSR [34], which proposed an Edge-oriented Convolutional Block (ECB) to achieve better performance, used relatively neat topology for low inference latency on mobile devices and introduced reparameterization techniques to achieve computation reduction for SR task. Following [34], RepSR [27] optimized the performance and training efficiency of the reparameterization module in ECBSR to achieve further computation reduction. ABPN [8] first attempted to apply operator substitution and used the faster *repeat* operator instead of nearest neighbor interpolation in global residual connections. Inspired by the reparameterization technique, we propose ET, which substitutes more types of operators to achieve low inference latency. Our proposed ETDS optimized the network from all three aspects, where ET optimized the network by achieving operator substitution and computation reduction while the dual stream network adopted a neat topology.

## 3. Methodology

In this section, we examine the current mobile super-resolution models in Sec. 3.1, and then propose a novel method, Equivalent Transformation (ET), which substitutes time-consuming operators with time-friendly ones in 3.2. However, applying ET comes at the cost of introducing redundant parameters. To address this, we carefully design a dual stream network that alleviates redundancy and enhances feature extraction ability in Sec. 3.3.

### 3.1. Analysis of Current Mobile SR Models

In order to design more efficient super-resolution models for mobile devices, we analyze the speed bottleneck of current mobile SR models on two of the most popular mobile phone SoCs[1]. As shown in Tab. 1, convolution and ReLU operations have lower inference latency than most other op-

[1] www.counterpointresearch.com/global-smartphone-ap-market-share/

Table 1. Average inference latency (ms) of common components in mobile SR models. The average inference latency of all components is tested on SR models with a channel size of 32, an input of size 360 $\times$ 640 and a scale factor of 3. For operators in the same type, the results of higher latency are marked in **bold**.

| Type | Operator | Latency (ms) | | |
|---|---|---|---|---|
| | | Dimensity 8100 | Snapdragon 870 | Snapdragon 865 |
| Convolution | $1 \times 1$ Convolution | 2.2 | 4 | 4 |
| | $3 \times 3$ Convolution | 10.4 | 18 | 21 |
| | Transposed Convolution | 7.3 | 7 | 9 |
| Activation | ReLU | 0.06 | 0.6 | 0.6 |
| | PReLU | **2.26** | **4.8** | **3.8** |
| | Leaky ReLU | **2.26** | **5.6** | **4.2** |
| | Sigmoid | 0.12 | 0.6 | 0.8 |
| | Tanh | 0.3 | 0.8 | 1 |
| Interpolation | Nearest neighbor | **26** | **24** | **27** |
| | Bilinear | **67** | **65** | **68** |
| Others | Add | 6.5 | 3 | 4 |
| | Concatenate | **22.8** | **6** | **7** |
| | Multiply | 6.5 | 3 | 4 |
| | Repeat | **16.3** | **19** | **14** |
| | Clip | **6.9** | **91** | **94** |
| | PixelShuffle | 4.2 | 3 | 4 |

erations, and PixelShuffle [23] is more efficient than transposed convolution, which indicates PixelShuffle and ReLU are better choices for mobile platforms. Hereafter, *plain* network refers to a network that only contains time-friendly operators (*i.e.*, convolution, ReLU activation functions and PixelShuffle), which has low inference latency on mobile devices.

To transform current mobile models into *plain* models, we may encounter the following challenges:

1) To avoid the rescaling problem in the INT8 quantization models, it is necessary to add the *clip* operator at the end of the model. However, the *clip* operator is also time-consuming (see Tab. 1), which requires a more efficient and equivalent alternative.
2) To compensate for the PSNR performance drop brought by the removal of global residual connections, it is necessary to find an equivalent way to transform the global residual connection into an efficient operator.

To tackle these challenges, we propose ET, which substitutes some time-consuming operators with time-friendly ones without reducing the reconstruction quality, thereby improving the running speed.

## 3.2. Equivalent Transformation

**Formal Definitions.** A convolution operation is formulated as $z = W \otimes x + b$, where $x \in \mathbb{R}^{c_i \times h \times w}$ and $z \in \mathbb{R}^{c_o \times h \times w}$ are the input and output tensors respectively, $W \in \mathbb{R}^{c_o \times k \times k \times c_i}$, $b \in \mathbb{R}^{c_o}$ are the kernel and the bias of the convolution, and $k$ is the kernel size, $c_i, c_o$ are the numbers of the input and output channels, and $h, w$ are the input height and width. $I$ is the identity kernel of the convolution operation, and $O$ is the kernel of zero elements, satisfying

$$I \otimes x = x, \quad O \otimes x = 0, \ \forall x \in \mathbb{R}^{c_i \times h \times w}. \qquad (1)$$

For input features and convolution kernels, the concatenation operation requires all but the concatenated dimension to be consistent. The channel dimension concatenation operation of feature vectors $x$ and $y$ is defined as $\begin{bmatrix} x \\ y \end{bmatrix}$. The concatenation operation of the convolution kernels $W_{h,1}$ and $W_{h,2}$ in the horizontal direction (*i.e.*, input-dimension concatenation) is defined as $\begin{bmatrix} W_{h,1}, W_{h,2} \end{bmatrix}$, which satisfies

$$\begin{bmatrix} W_{h,1}, W_{h,2} \end{bmatrix} \otimes \begin{bmatrix} x \\ y \end{bmatrix} = W_{h,1} \otimes x + W_{h,2} \otimes y. \qquad (2)$$

Likewise, the concatenation operation of the convolution kernels $W_{v,1}$ and $W_{v,2}$ in the vertical direction, *i.e.*, output-dimension concatenation is defined as $\begin{bmatrix} W_{v,1} \\ W_{v,2} \end{bmatrix}$, which satisfies

$$\begin{bmatrix} W_{v,1} \\ W_{v,2} \end{bmatrix} \otimes x = \begin{bmatrix} W_{v,1} \otimes x \\ W_{v,2} \otimes x \end{bmatrix}. \qquad (3)$$

More details will be found in Appendix.

**ET for *Repeat* Operator.** Given an input tensor $x \in \mathbb{R}^{c \times h \times w}$, the *repeat* operator *repeat*$(x, n)$ constructs a new tensor $z \in \mathbb{R}^{(nc) \times h \times w}$ by replicating the input $n$ times along the channel dimension. This operation can be expressed as a convolution layer. Precisely, Eq. (1) and (3) yield

$$\begin{aligned} \textit{repeat}(x, n) &= \begin{bmatrix} x \\ \vdots \\ x \end{bmatrix} = \begin{bmatrix} I \otimes x \\ \vdots \\ I \otimes x \end{bmatrix} = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix} \otimes x \\ &= \textit{repeat}(I, n) \otimes x, \end{aligned} \qquad (4)$$

where *repeat*$(I, n)$ is a constant. Thus, the *repeat* operator is equivalently transformed into a convolution layer, whose kernel and bias are *repeat*$(I, n)$ and a zero vector, respectively.

**ET for *Add* Operator.** The *add* operator refers to the element-wise add operation of two feature vectors and is an indispensable part of residual connections, *i.e.*, $z = x + y$. From Eq. (1) and (2), we have:

$$x + y = I \otimes x + I \otimes y = \begin{bmatrix} I, I \end{bmatrix} \otimes \begin{bmatrix} x \\ y \end{bmatrix}. \qquad (5)$$

Thus, the *add* operator is equivalently transformed into a convolution layer with a *concatenate* operator, whose kernel and bias are $\begin{bmatrix} I, I \end{bmatrix}$ and a zero vector, respectively. Notably, this proposed convolution layer can be eliminated later by reparameterization techniques.

**ET for *Concatenate* Operator.** In our model, the *concatenate* operator always follows the convolution layers.

We integrate the convolution-concatenate structure, *i.e.*, the *concatenate* operator and its preceding convolution layer, which can be simplified with Eq. (2) and (3) as follows:

$$z = \begin{bmatrix} W_1 \otimes x + b_1 \\ W_2 \otimes y + b_2 \end{bmatrix} = \begin{bmatrix} W_1 & O \\ O & W_2 \end{bmatrix} \otimes \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \quad (6)$$

Following Eq. (6), the convolution-concatenate structure is transformed into a concatenate-convolution structure, with the kernel and bias as follows:

$$\begin{bmatrix} W_1 & O \\ O & W_2 \end{bmatrix} \text{ and } \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \quad (7)$$

In particular, when the residual branch does not have a convolution layer, it can be expressed as:

$$z = \begin{bmatrix} W \otimes x + b \\ y \end{bmatrix} = \begin{bmatrix} W & O \\ O & I \end{bmatrix} \otimes \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b \\ 0 \end{bmatrix}. \quad (8)$$

When $x = y$, Eq. (8) can be further rewritten as:

$$z = \begin{bmatrix} W \otimes x + b \\ x \end{bmatrix} = \begin{bmatrix} W \\ I \end{bmatrix} \otimes x + \begin{bmatrix} b \\ 0 \end{bmatrix}, \quad (9)$$

where the kernel is $\begin{bmatrix} W \\ I \end{bmatrix}$ and thus the *concatenate* operator is eliminated.

**ET for *Clip* Operator.** The *clip* operator is expressed as:

$$clip(x) = \begin{cases} 0, & x \leq 0, \\ x, & 0 < x \leq 255, \\ 255, & x > 255, \end{cases} \quad (10)$$

which avoids the loss of precision and subsequent additional inverse scaling operations caused by mismatched normalization of the INT8 quantization model [12]. Nevertheless, as shown in Tab. 1, the *clip* operator is extremely time-consuming on many mobile SoCs, so it is of great necessity to find a time-friendly substitution for the *clip* operator.

To equivalently transform the *clip* operator with ReLU , we reformulate *clip* operator as:

$$clip(x) = \mathrm{ReLU}(-\mathrm{ReLU}(-x + 255) + 255). \quad (11)$$

To speed up this process, we further rewrite it as:

$$\begin{cases} y = & \mathrm{ReLU}(-I \otimes x + 255), \\ clip(x) = & \mathrm{ReLU}(-I \otimes y + 255). \end{cases} \quad (12)$$

Using Eq. (12), the *clip* operator is transformed into two convolutions with ReLU, whose kernels are $-I$ and bias is a vector with all elements equal to 255. Remark that the first convolution layer in Eq. (12) is eliminated by reparameterization techniques [5].

**ET for Equivalent Plain Model Conversion.** As depicted in Fig. 2, the model is converted into an equivalent plain model by ET as following:

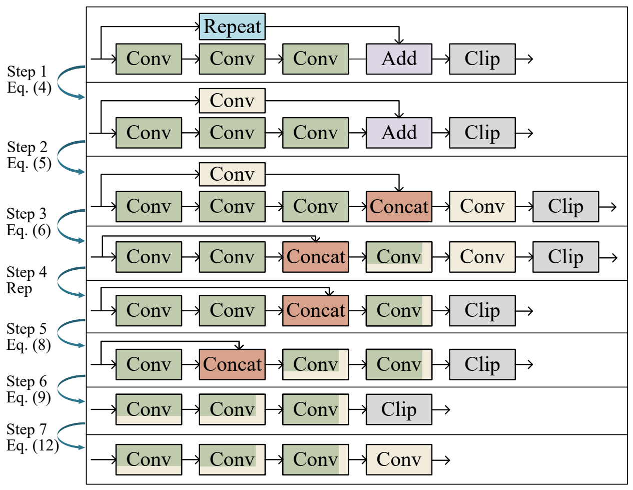

Figure 2. Illustration of converting the general model into the equivalent plain model using ET. The beige convolution indicates that the convolution kernel is assigned a specific value. Rep refers to the reparameterization technique.

- The *repeat* operator and the *add* operator are converted into a single convolution and a *concatenate* operator respectively (see Step 1 and Step 2).
- The *concatenate* operator with the preceding convolutions from two branches is converted into a convolution. Notably, the global residual connection does not contain negative values, so the ReLU function is omitted (see Step 3).
- The two convolutions introduced by previous steps are converted into a convolution using reparameterization techniques (see Step 4).
- The convolution-concatenate structure is converted into a concatenate-convolution one, thus the *concatenate* operator is converted and moves forward till it reaches the skip connection starting point, that is, $x$ equals $y$. In this situation, the *concatenate* operator is further eliminated (see Step 5 and Step 6).
- The *clip* operator is converted into the convolution (see Step 7).

**Remark 1** *Although ET speeds up inference by transforming time-consuming operators into time-friendly ones, it introduces some redundant parameters. For instance, in the transformed convolution kernel* $\mathrm{diag}(W, I)$ *derived from Eq. (8). All learnable parameters appear in* $W$. *If the redundant parameters are effectively utilized, better reconstruction performance can be achieved.*

## 3.3. Dual Stream Network

To alleviate the side effects of the redundant parameters of ET, we design a dual stream network where the two branches learn the low-frequency and high-frequency contents of the image separately and utilize the parameters to

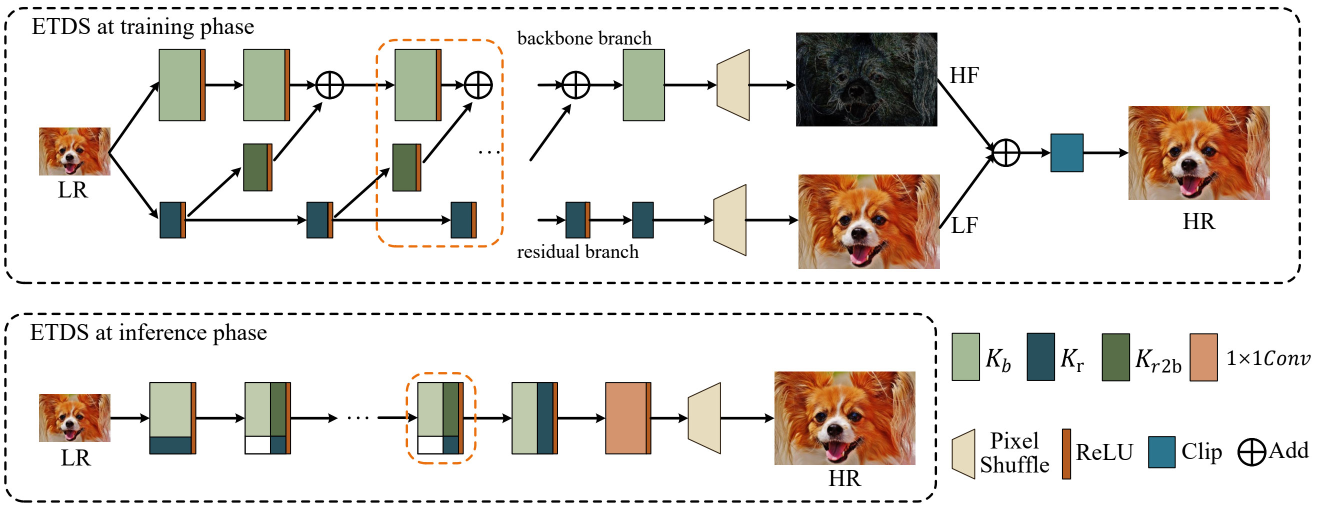

Figure 3. Network architecture of ETDS during training and inference stages. The residual branch for extracting low-frequency contents only contains a few parameters, and most of the parameters are used to extract high-frequency contents.

the fullest. The transformed convolution kernel in Eq. (8) is $\begin{bmatrix} K_b & K_{r2b} \\ K_{b2r} & K_r \end{bmatrix}$, where $K_b$ and $K_r$ denote the modules that constitute the backbone branch and residual branch, respectively. $K_{r2b}$ is the module that transfers information from the residual branch to the backbone branch, and $K_{b2r}$ transfers information reversely. The residual branch consisting of $K_r$ and the backbone branch consisting of $K_b$ extract low-frequency and high-frequency information respectively.

Initially, $K_r$, $K_{r2b}$ and $K_{b2r}$ are not learnable. Furthermore, we have analyzed the the possible impacts of making them learnable on the reconstruction quality and found that when $K_r$ and $K_{r2b}$ are learnable, they can extract low-frequency information more effectively and transfer more residual information to the backbone branch, respectively. However, when $K_{b2r}$ is learnable, the backbone branch may take less important low-frequency information into consideration, which reduces the number of parameters for extracting high-frequency components, and potentially compromising the performance.

As shown in Fig. 3, $K_r$ and $K_{r2b}$ in our dual stream network contain learnable parameters, while parameters in $K_{b2r}$ remain fixed. In the training phase, we encourage the residual branch to learn as many low-frequency contents as possible, that is, to minimize the $L_1$ distance between the output of the residual branch ($I_{LF}$) and the ground truth ($I_{GT}$):

$$\mathcal{L}_{LF} = \|I_{GT} - I_{LF}\|_1. \tag{13}$$

The backbone branch learns high-frequency contents, and its output ($I_{HF}$) compensates for the gap between $I_{GT}$ and $I_{LF}$:

$$\mathcal{L}_{HF} = \|I_{GT} - I_{LF} - I_{HF}\|_1. \tag{14}$$

To train the dual stream network, we use the overall loss function:

$$\mathcal{L}_{DS} = \mathcal{L}_{HF} + \alpha \times \mathcal{L}_{LF}, \tag{15}$$

which drives more parameters to extract high-frequency contents to improve the overall performance. In the inference stage, as shown in Fig. 3, the dual stream network can be equivalently transformed into a *plain* network, which ensures low inference latency.

## 4. Experiments

### 4.1. Datasets and Metrics

Our model ETDS is trained on DIV2K dataset [25] with 800 training images. The performance is evaluated on 5 benchmark datasets (*i.e.*, Set5 [2], Set14 [32], BSDS100 [21], Urban100 [9] and DIV2K100 [25]). PSNR and SSIM [29] are used to evaluate the reconstruction quality and calculated on the Y channel in the YCbCr space.

### 4.2. Implementation Details

ETDS is implemented on PyTorch-based [22] BasicSR [28] framework and trained on an NVIDIA 2080Ti GPU. In the training phase, 32 patches of $64 \times 64$ LR image are used as input. Random rotation, horizontal flipping, and channel shuffle are applied for data augmentation. We train our model by the ADAM [14] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We set $\mathcal{L}_{\mathrm{DS}}$ as the loss function with $\alpha = 0.5$. In accordance with previous works, our model is trained for $1600k$ iterations with a learning rate of $5e^{-4}$. For the first $100k$ iterations, $K_r$ and $K_{r2b}$ are initialized by setting $K_r = I$ and $K_{r2b} = O$. In the testing phase, we test the inference latency on three SoCs (*i.e.*, Dimensity 8100, Snapdragon 888, and Snapdragon 8 Gen 1) with the

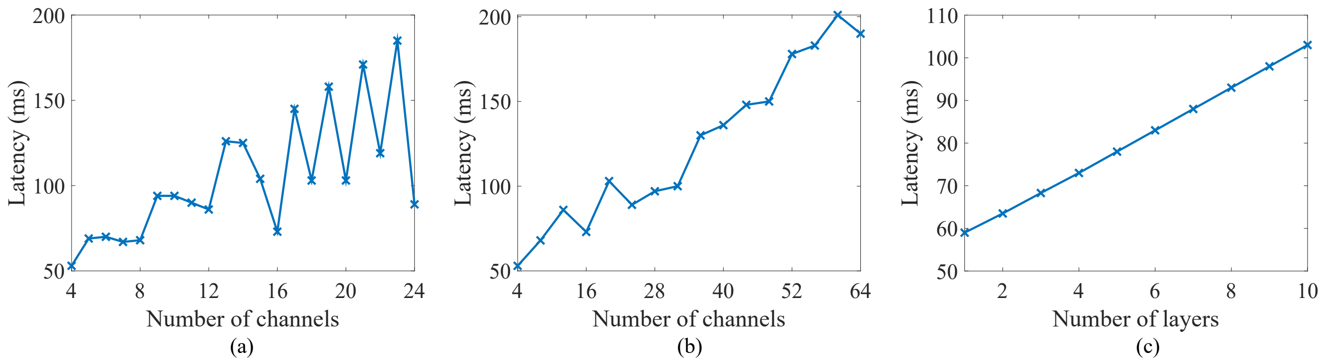

Figure 4. The effect of the numbers of channels and layers on inference latency of the plain model. (a) shows the inference latency with the number of channels between 4 and 24. (b) shows the inference latency with the number of channels between 4 and 64, divisible by 4. (c) shows the inference latency with the number of layers between 1 and 10.

AI Benchmark application [11, 13]. All inference latencies and multiply-accumulate operations (MACs) are tested with inputs of $360 \times 640$.

## 4.3. Hardware-Friendly Number of Channels

As shown in Fig. 4, the numbers of channels and layers have a significant impact on the model's efficiency. Observing Fig. 4 (a)(b), we can see that the models have lower inference latency on mobile devices when the number of channels is a multiple of 4, due to the processing unit alignment [31], the memory alignment, and other hardware optimizations [4]. Fig. 4 (c) supports that the inference latency is proportional to the number of layers.

As a result, for a lower inference latency, ETDS is designed with 16, 32, and 48 channels and the number of layers in the model is designed for balancing the performance and inference latency. Also, for the equivalently transformed models (*e.g.*, ECBSR+ET and ABPN+ET), extra channels are added to ensure that the number of channels is a multiple of 4.

## 4.4. Comparisons to Previous Works

We compare our ETDS with representative real-time SR models on mobile devices, including Bicubic, ESPCN [23], FSRCNN [7], ABPN [8], and ECBSR [34]. Since ESPCN, FSRCNN, and ECBSR only process on a single channel of the input image, some preprocessing and postprocessing, such as color space conversion and upsampling of the other two channels are omitted. For a fair comparison, these models are retrained in the RGB color space.

We train four different sizes of our model, denoted as ETDS-T, ETDS-S, ETDS-M and ETDS-L for different computational use cases, with channels 16, 32, 32 and 48, and layers 5, 4, 6, and 7, respectively. The performance comparison is summarized in Tab. 2. In addition to PSNR/SSIM indexes, we also list the number of parameters and MACs for a more comprehensive comparison.

**Quantitative Results.** The quantitative comparison is presented in Tab. 2. From Tab. 2, our model achieves not only lower inference latency than other models in most cases but also higher PSNR/SSIM indexes. For instance, as shown in Tab. 2 (latency column i), our ETDS-S runs 1.2ms faster than ECBSR-M4C16 on the $\times 3$ task, and its PSNR index on the Urban100 dataset is 0.29dB higher. ETDS-L's inference latency on the $\times 3$ task is 3.5ms faster than the ABPN-M6C40's, and PSNR on Set5. Notably, although our model consists of slightly more parameters and MACs, it is still faster than the state-of-the-art models. This phenomenon occurs due to two-fold aspects. 1). Under different contexts, the same operations may have different inference latency. 2). Some operations are not measured in parameters and MACs metrics. For instance, ReLU does not influence parameters and *concatenation* does not influence MACs.

**Qualitative Results.** The qualitative comparison results are depicted in Fig. 5. We can see that the texture of the wall in thimagese images generated by ESPCN [23] and FSRCNN [7] is blurred, and images generated by ECBSR-M6C40 [34] and ABPN-M6C40 [8] are slightly distorted. Obviously, our ETDS-T produces the least distorted and blurred images with the best PSNR/SSIM results.

## 4.5. Ablation Studies

In this section, we propose a series of ablation studies on the design of the proposed ETDS. All the ablation experiments are conducted on the $\times 4$ task. We report PSNR values on the Y channel in the YCbCr space.

**Ablation Studies for Equivalent Transformation.** We validate the effectiveness of ET through a series of ablation studies. Additionally, we apply ET to both ECBSR and ABPN to further demonstrate its effectiveness. Note that ETDS does not have a repeat operator and we do not apply ET for the concatenate operator for ECBSR and ABPN because the improvment on the NNAPI driver is insignificant. Our results in Tab. Tab. 3 show that applying ET to these operators results in reduced latency.

Table 2. Performance comparison of different SR models on five benchmark datasets. PSNR/SSIM on the Y channel is reported on each dataset. Inference latency i, ii, iii and iv denotes the inference latency of the models using NNAPI driver on Dimensity 8100 SoC, MediaTek Neuron driver on Dimensity 8100 SoC, Qualcomm QNN GPU driver on Snapdragon 888 SoC, and NNAPI on Snapdragon 8 Gen 1 SoC respectively.

| Scale | Model | Params (K) | MACs (G) | Latency (ms) | | | | Set5 | Set14 | B100 | Urban100 | DIV2K100 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | i | ii | iii | iv | PSNR/SSIM | PSNR/SSIM | PSNR/SSIM | PSNR/SSIM | PSNR/SSIM |
| ×2 | Bicubic | * | * | * | * | * | * | 33.95/0.9334 | 30.54/0.8751 | 29.74/0.8503 | 27.07/0.8456 | 32.67/0.9078 |
| | ESPCN | 26.80 | 6.15 | 6.30 | 4.83 | **91** | - | 36.87/0.9559 | 32.62/0.9086 | 31.40/0.8898 | 29.61/0.8973 | 34.78/0.9347 |
| | ETDS-T (Ours) | 13.94 | 3.19 | **4.85** | **3.81** | 125 | - | **37.18/0.9578** | **32.76/0.9103** | **31.62/0.8930** | **30.10/0.9058** | **35.09/0.9375** |
| | FSRCNN | 24.68 | 15.05 | 10.4 | 5.89 | 414 | 562 | 37.00/0.9568 | 32.68/0.9094 | 31.51/0.8913 | 29.92/0.9022 | 34.97/0.9363 |
| | ECBSR-M4C16 | 11.55 | 2.64 | **6.15** | **4.57** | 189 | 31.6 | 37.18/0.9577 | 32.79/0.9105 | 31.61/0.8928 | 30.18/0.9064 | 35.10/0.9375 |
| | ETDS-S (Ours) | 41.51 | 9.52 | 8.06 | 5.77 | **169** | - | **37.38/0.9587** | **32.96/0.9124** | **31.77/0.8951** | **30.62/0.9121** | **35.36/0.9396** |
| | ECBSR-M4C32 | 41.52 | 9.53 | **9.70** | **6.64** | 372 | 31.4 | 37.39/0.9587 | 33.03/0.9126 | 31.79/0.8953 | 30.69/0.9128 | 35.39/0.9398 |
| | ABPN-M4C28 | 33.46 | 7.67 | 10.0 | 6.66 | **207** | 28.5 | 37.36/0.9585 | 32.96/0.9121 | 31.75/0.8949 | 30.55/0.9111 | 35.31/0.9392 |
| | ETDS-M (Ours) | 60.01 | 13.77 | 10.2 | 6.73 | 288 | - | **37.54/0.9593** | **33.09/0.9133** | **31.86/0.8963** | **30.87/0.9149** | **35.50/0.9406** |
| | ECBSR-M6C40 | 92.37 | 21.22 | **18.1** | **10.9** | 653 | 46.2 | 37.61/0.9596 | 33.18/0.9139 | 31.94/0.8972 | 31.09/0.9174 | 35.61/0.9415 |
| | ABPN-M6C40 | 93.40 | 21.45 | 18.7 | 11.2 | **435** | 31.9 | 37.58/0.9594 | 33.16/0.914 | 31.92/0.8971 | 31.04/0.9169 | 35.59/0.9414 |
| | ETDS-L (Ours) | 152.18 | 34.97 | 20.6 | 12.1 | 469 | - | **37.64/0.9597** | **33.24/0.9145** | **31.98/0.8977** | **31.22/0.9188** | **35.69/0.9421** |
| ×3 | Bicubic | * | * | * | * | * | * | 30.61/0.8719 | 27.79/0.7811 | 27.31/0.7453 | 24.58/0.7396 | 29.80/0.8341 |
| | ESPCN | 31.13 | 7.14 | 9.79 | 8.38 | **110** | **44** | 32.85/0.9115 | 29.41/0.8236 | 28.40/0.7876 | 26.22/0.8002 | 31.24/0.8670 |
| | ETDS-T (Ours) | 16.92 | 3.86 | **8.64** | **5.36** | 140 | 49.2 | **33.14/0.9160** | **29.53/0.8274** | **28.54/0.7920** | **26.48/0.8111** | **31.43/0.8713** |
| | FSRCNN | 24.68 | 30.73 | 13.7 | **7.32** | 456 | 1350 | 33.01/0.9145 | 29.52/0.8264 | 28.50/0.7903 | 26.40/0.8074 | 31.38/0.8699 |
| | ECBSR-M4C16 | 13.72 | 3.14 | 16.2 | 11.0 | 216 | 69.4 | 33.07/0.9149 | 29.53/0.8269 | 28.57/0.7921 | 26.61/0.8136 | 31.44/0.8712 |
| | ETDS-S (Ours) | 46.79 | 10.73 | **12.2** | 7.73 | **185** | **49.2** | **33.49/0.9194** | **29.76 0.8314** | **28.71/0.7961** | **26.90/0.8221** | **31.69/0.8756** |
| | ECBSR-M4C32 | 45.85 | 10.52 | 19.7 | 13.1 | 354 | 69.2 | 33.56/0.9205 | 29.78/0.8319 | 28.73/0.7965 | 26.98/0.8239 | 31.73/0.8761 |
| | ABPN-M4C28 | 42.54 | 9.76 | 20.5 | 13.8 | 275 | 62.8 | 33.53/0.9200 | 29.78/0.8317 | 28.72/0.7966 | 26.94/0.8232 | 31.71/0.8758 |
| | ETDS-M (Ours) | 65.29 | 14.98 | **14.4** | **8.76** | **260** | **49** | **33.65/0.9214** | **29.86/0.8333** | **28.78/0.7981** | **27.12/0.8278** | **31.81/0.8776** |
| | ECBSR-M6C40 | 97.79 | 22.46 | 28.1 | 17.4 | 713 | 75.3 | 33.67/0.9211 | 29.92/0.8339 | 28.85/0.7994 | 27.36/0.8333 | 31.88/0.8784 |
| | ABPN-M6C40 | 104.10 | 23.91 | 29.2 | 18.2 | 480 | 66 | 33.71/0.9213 | 29.92/0.8339 | 28.85/0.7994 | 27.37/0.8336 | 31.89/0.8786 |
| | ETDS-L (Ours) | 159.77 | 36.71 | **25.4** | **14.2** | **444** | **51.8** | **33.88/0.9235** | **30.00/0.8359** | **28.90/0.8010** | **27.45/0.8359** | **32.00/0.8807** |
| ×4 | Bicubic | * | * | * | * | * | * | 28.60/0.8140 | 26.21/0.7088 | 26.04/0.6733 | 23.23/0.6613 | 28.23/0.7775 |
| | ESPCN | 37.20 | 8.54 | 11.8 | 8.14 | **120** | 79.8 | 30.66/0.8688 | 27.66/0.7581 | 26.94/0.7152 | 24.56/0.7263 | 29.46/0.8133 |
| | ETDS-T (Ours) | 21.36 | 4.88 | **10.9** | **6.94** | 163 | **77.4** | **30.87/0.8738** | **27.75/0.7618** | **27.04/0.7194** | **24.71/0.7350** | **29.59/0.8172** |
| | FSRCNN | 24.68 | 52.68 | - | - | - | - | 30.76/0.8721 | 27.74/0.7609 | 27.01/0.7181 | 24.66/0.7330 | 29.56/0.8163 |
| | ECBSR-M4C16 | 16.77 | 3.83 | 23.9 | 13.7 | 278 | 191 | 30.87/0.8741 | 27.80/0.7626 | 27.06/0.7195 | 24.75/0.7364 | 29.61/0.8176 |
| | ETDS-S (Ours) | 54.11 | 12.41 | **14.6** | **9.02** | **203** | **77.4** | **31.19/0.8806** | **28.01/0.7678** | **27.18/0.7240** | **25.03/0.7479** | **29.80/0.8224** |
| | ECBSR-M4C32 | 51.92 | 11.91 | 28.0 | 16.3 | 399 | 191 | 31.24/0.8815 | 28.04/0.7686 | 27.21/0.7248 | 25.07/0.7494 | 29.84/0.8231 |
| | ABPN-M4C28 | 62.05 | 14.24 | 30.3 | 17.4 | 335 | 187 | 31.32/0.8833 | 28.10/0.7697 | 27.23/0.7254 | 25.12/0.7511 | 29.86/0.8237 |
| | ETDS-M (Ours) | 72.61 | 16.66 | **16.6** | **9.98** | **267** | **78.9** | **31.41/0.8843** | **28.13/0.7705** | **27.27/0.7265** | **25.20/0.7544** | **29.92/0.8250** |
| | ECBSR-M6C40 | 105.37 | 24.20 | 36.8 | 20.7 | 749 | 195 | 31.41/0.8831 | 28.20/0.7716 | 27.33/0.7280 | 25.37/0.7601 | 29.97/0.8259 |
| | ABPN-M6C40 | 125.87 | 28.91 | 39.1 | 22.4 | 538 | 191 | 31.62/0.8876 | 28.27/0.7737 | 27.35/0.7293 | 25.41/0.7616 | 30.04/0.8277 |
| | ETDS-L (Ours) | 169.97 | 39.05 | **27.6** | **15.3** | **447** | **85.3** | **31.69/0.8889** | **28.31/0.7751** | **27.37/0.7302** | **25.47/0.7643** | **30.09/0.8289** |

Table 3. Ablation operation for each operator. All indicators are tested using NNAPI driver on Dimensity 8100 SoC.

| Model | Size | Improvement | | |
|---|---|---|---|---|
| | | add | concatenate | clip |
| ETDS | T | 21.03% | 28.92% | 0.26% |
| | S | 16.74% | 26.61% | 0.86% |
| | M | 8.33% | 34.47% | 0.76% |
| | L | 7.29% | 27.64% | 0.50% |

| | | repeat | add | clip |
|---|---|---|---|---|
| ECBSR | M4C16 | 47.95% | 7.38% | 1.23% |
| | M4C32 | 39.01% | 11.35% | -0.35% |
| | M6C40 | 29.92% | 10.24% | 0.54% |
| ABPN | M4C28 | 37.46% | 10.75% | 1.30% |
| | M6C40 | 27.30% | 10.20% | 0.77% |

**Ablation Studies for Global Residual Connection.** Here we investigate the effect of the global residual connection. Tab. 4 shows the results of the experiments. The PSNR value is improved by 0.04dB on Set5, when adding a global residual connection composed of nearest neighbor interpolation to Plain-M6C32.

Table 4. PSNR/SSIM results for different variants of ETDS-M.

| Variant | Modifications | Set5 | Urban100 |
|---|---|---|---|
| | | PSNR/SSIM | PSNR/SSIM |
| I | Plain-M6C32 | 31.32/0.8830 | 25.10/0.7505 |
| II | Plain-M6C32 w/ interpolation | 31.36/0.8833 | 25.13/0.7519 |
| III | ETDS-M w/o $K_{r2b}$ module | 31.36/0.8835 | 25.16/0.7526 |
| IV | ETDS-M | **31.41/0.8843** | **25.20/0.7544** |
| V | ETDS-M w/ $K_{b2r}$ module | 31.38/0.8837 | 25.17/0.7530 |
| VI | ETDS-M w/o $\mathcal{L}_{DS}$ | 31.35/0.8837 | 25.14/0.7521 |

**Ablation Studies for $K_r$, $K_{r2b}$ and $K_{b2r}$.** We train $K_r$, $K_{r2b}$, and $K_{b2r}$ one after another in an incremental way. The variants are recorded as Variant III, Variant IV, and Variant V in Tab. 4. Variant IV refers to our ETDS-M. From Tab. 4, the performance of Variant II, III and IV increases sequentially, indicating that the learnability of $K_r$ and $K_{r2b}$ is beneficial to improve the performance. However, Variant V's performance is worse than Variant IV's, which may indicate that introducing a learnable $K_{b2r}$ could lead to a degradation in performance. The $K_{b2r}$ module connects the low-frequency constraints to the backbone branch during backpropagation which causes the backbone branch to

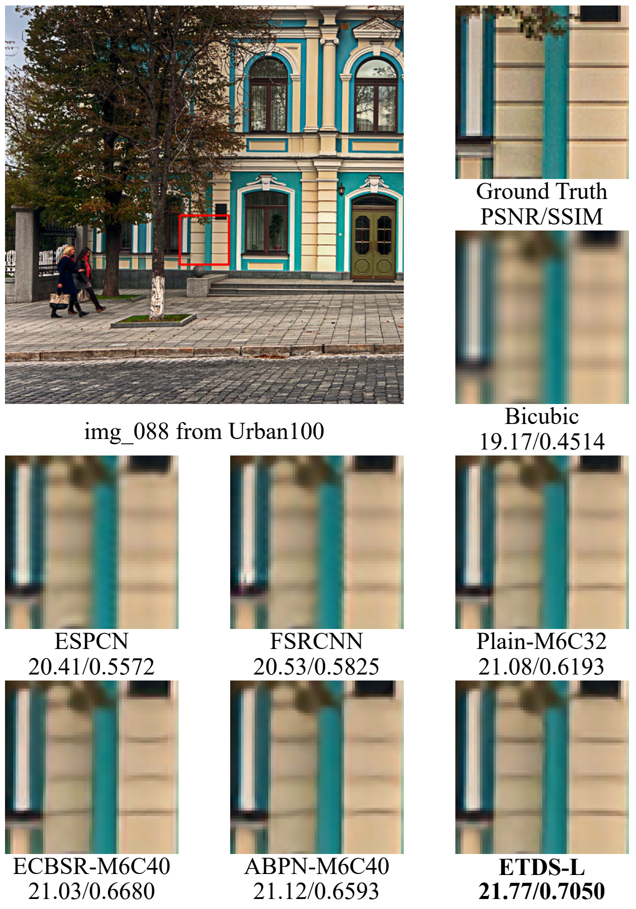

Figure 5. Qualitative comparison of real-time SR models on Set5 for ×4 upscaling task.

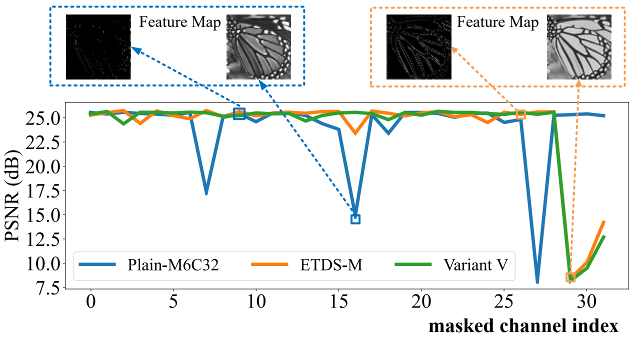

Figure 6. Channel masking experiments with Plain-M6C32 model, ETDS and Variant V. In order to understand the meaning of each channel, the visualization images of some channels are shown in this figure.

pay extra attention to low-frequency information, thereby reducing the model's ability to extract high-frequency information.

**Ablation Studies for Dual Stream Constraint $\mathcal{L}_{\text{DS}}$.** For ETDS-M, the PSNR value is 0.06dB higher than the one without $\mathcal{L}_{\text{DS}}$ (Tab. 4 Variant VI), which supports the benefit of the dual stream constraint $\mathcal{L}_{\text{DS}}$.

**Analysis of Dual Stream Network.** We conduct the channel mask experiments [16] on ETDS-M, Plain-M6C32 and Variant V models, and then analyze how each channel contributes to the performance. From Fig. 6, the low-frequency information of ETDS-M is mainly concentrated in the last three channels (*i.e.*, the residual branch), while the high-frequency information is distributed in the remaining channels (*i.e.*, the backbone branch) corresponding to our design in Sec. 3.3. In addition, the extraction of the low-frequency channel of ETDS-M is more effective than the Plain-M6C32 model. This may be the reason why ETDS-M has better performance than the Plain-M6C32 model. On top of that, Variant V does better in extracting low-frequency information than ETDS-M, but worse in PSNR performance, which confirms that the learnability of $K_{b2r}$ compromises the ability to extract high-frequency information.

## 4.6. Limitations

While our method achieves a better trade-off between performance and latency on mobile devices, it also has its limitations. Firstly, ET is only appropriate for relatively simple network architectures. More complex structures would require a more complicated ET process, which could potentially introduce an excessive number of redundant parameters. Secondly, due to its simplified structure, very deep networks still pose a challenge for ETDS, and its performance may not reach the level of its EDSR counterpart.

## 5. Conclusion

This paper has presented ETDS for mobile image super-resolution via Equivalent Transformation and dual stream network construction. ET method is applied to substitute time-consuming operators, including *add*, *repeat*, *clip* and *concatenate*, with time-friendly ones (*convolution* and ReLU), which can alleviate the inference latency. We have developed a dual stream network, which successfully eliminated most of the redundant parameters brought by the implementation of ET. Our approach represents an innovative integration of the ET method with a dual-stream network for mobile devices. Extensive experiments have demonstrated that ETDS can outperform the state-of-the-art lightweight SR models in terms of inference latency and reconstruction quality.

**Acknowledgements.** This work was supported in part by "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software (Grant No. 22510750100), and Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

## References

[1] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X*, volume 11214 of *Lecture Notes in Computer Science*, pages 256–272. Springer, 2018. 2

[2] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie-Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In Richard Bowden, John P. Collomosse, and Krystian Mikolajczyk, editors, *British Machine Vision Conference, BMVC 2012, Surrey, UK, September 3-7, 2012*, pages 1–10. BMVA Press, 2012. 1, 5

[3] Xiangyu Chen, Xintao Wang, Jiantao Zhou, and Chao Dong. Activating more pixels in image super-resolution transformer. *CoRR*, abs/2205.04437, 2022. 1, 2

[4] John Cheng, Max Grossman, and Ty McKercher. *Professional CUDA c programming*. John Wiley & Sons, 2014. 6

[5] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 13733–13742. Computer Vision Foundation / IEEE, 2021. 4

[6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *CoRR*, abs/1501.00092, 2015. 1, 2

[7] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, volume 9906 of *Lecture Notes in Computer Science*, pages 391–407. Springer, 2016. 6

[8] Zongcai Du, Jie Liu, Jie Tang, and Gangshan Wu. Anchor-based plain net for mobile image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*, pages 2494–2502. Computer Vision Foundation / IEEE, 2021. 1, 2, 6

[9] 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, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 5197–5206. IEEE Computer Society, 2015. 5

[10] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In Laurent Amsaleg, Benoit Huet, Martha A. Larson, Guillaume Gravier, Hayley Hung, Chong-Wah Ngo, and Wei Tsang Ooi, editors, *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*, pages 2024–2032. ACM, 2019. 2

[11] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. AI benchmark: Running deep neural networks on android smartphones. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part V*, volume 11133 of *Lecture Notes in Computer Science*, pages 288–314. Springer, 2018. 6

[12] Andrey Ignatov, Radu Timofte, Maurizio Denna, and Abdel Younes. Real-time quantized image super-resolution on mobile npus, mobile AI 2021 challenge: Report. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*, pages 2525–2534. Computer Vision Foundation / IEEE, 2021. 4

[13] Andrey Ignatov, Radu Timofte, Andrei Kulik, Seungsoo Yang, Ke Wang, Felix Baum, Max Wu, Lirong Xu, and Luc Van Gool. AI benchmark: All about deep learning on smartphones in 2019. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pages 3617–3635. IEEE, 2019. 6

[14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5

[15] Fangyuan Kong, Mingxi Li, Songwei Liu, Ding Liu, Jingwen He, Yang Bai, Fangmin Chen, and Lean Fu. Residual local feature network for efficient super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2022, New Orleans, LA, USA, June 19-20, 2022*, pages 765–775. IEEE, 2022. 2

[16] Xiangtao Kong, Xina Liu, Jinjin Gu, Yu Qiao, and Chao Dong. Reflash dropout in image super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5992–6002. IEEE, 2022. 8

[17] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, BC, Canada, October 11-17, 2021*, pages 1833–1844. IEEE, 2021. 1, 2

[18] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1132–1140. IEEE Computer Society, 2017. 1, 2

[19] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In Adrien Bartoli and Andrea Fusiello, editors, *Computer Vision - ECCV 2020 Workshops - Glasgow, UK, August 23-28, 2020, Proceedings, Part III*, volume 12537 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2020. 2

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 9992–10002. IEEE, 2021. 2

[21] David R. Martin, Charless C. Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01), Vancouver, British Columbia, Canada, July 7-14, 2001 - Volume 2*, pages 416–425. IEEE Computer Society, 2001. 5

[22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019. 5

[23] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1874–1883. IEEE Computer Society, 2016. 3, 6

[24] Wuzhen Shi, Feng Jiang, and Debin Zhao. Single image super-resolution with dilated convolution based multi-scale information learning inception module. *CoRR*, abs/1707.07128, 2017. 2

[25] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. NTIRE 2017 challenge on single image super-resolution: Methods and results. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1110–1121. IEEE Computer Society, 2017. 5

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. 2

[27] Xintao Wang, Chao Dong, and Ying Shan. Repsr: Training efficient vgg-style super-resolution networks with structural re-parameterization and batch normalization. *CoRR*, abs/2205.05671, 2022. 1, 2

[28] Xintao Wang, Liangbin Xie, Ke Yu, Kelvin C.K. Chan, Chen Change Loy, and Chao Dong. BasicSR: Open source image and video restoration toolbox. https://github.com/XPixelGroup/BasicSR, 2022. 5

[29] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004. 5

[30] Kailu Wu, Chung-Kuei Lee, and Kaisheng Ma. Memsr: Training memory-efficient lightweight model for image super-resolution. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 24076–24092. PMLR, 2022. 2

[31] Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1803–1811. IEEE, 2019. 6

[32] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In Jean-Daniel Boissonnat, Patrick Chenin, Albert Cohen, Christian Gout, Tom Lyche, Marie-Laurence Mazure, and Larry L. Schumaker, editors, *Curves and Surfaces - 7th International Conference, Avignon, France, June 24-30, 2010, Revised Selected Papers*, volume 6920 of *Lecture Notes in Computer Science*, pages 711–730. Springer, 2010. 5

[33] Dafeng Zhang, Feiyu Huang, Shizhuo Liu, Xiaobing Wang, and Zhezhu Jin. Swinfir: Revisiting the swinir with fast fourier convolution and improved training for image super-resolution. *CoRR*, abs/2208.11247, 2022. 1

[34] Xindong Zhang, Hui Zeng, and Lei Zhang. Edge-oriented convolution block for real-time super resolution on mobile devices. In Heng Tao Shen, Yueting Zhuang, John R. Smith, Yang Yang, Pablo Cesar, Florian Metze, and Balakrishnan Prabhakaran, editors, *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*, pages 4034–4043. ACM, 2021. 1, 2, 6

[35] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, volume 11211 of *Lecture Notes in Computer Science*, pages 294–310. Springer, 2018. 1, 2

[36] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2472–2481. Computer Vision Foundation / IEEE Computer Society, 2018. 1, 2