Transformer for Single Image Super-Resolution

Zhisheng Lu[1†], Juncheng Li[2†], Hong Liu[1*], Chaoyan Huang[3], Linlin Zhang[1], Tieyong Zeng[2]
[1]Peking University Shenzhen Graduate School [2]The Chinese University of Hong Kong
[3]Nanjing University of Posts and Telecommunications
{zhisheng_lu, hongliu, catherinezll}@pku.edu.cn
cvjunchengli@gmail.com, Huangchy2020@163.com, zeng@math.cuhk.edu.hk

Abstract

Single image super-resolution (SISR) has witnessed great strides with the development of deep learning. However, most existing studies focus on building more complex networks with a massive number of layers. Recently, more and more researchers start to explore the application of Transformer in computer vision tasks. However, the heavy computational cost and high GPU memory occupation of the vision Transformer cannot be ignored. In this paper, we propose a novel Efficient Super-Resolution Transformer (ESRT) for SISR. ESRT is a hybrid model, which consists of a Lightweight CNN Backbone (LCB) and a Lightweight Transformer Backbone (LTB). Among them, LCB can dynamically adjust the size of the feature map to extract deep features with a low computational costs. LTB is composed of a series of Efficient Transformers (ET), which occupies a small GPU memory occupation, thanks to the specially designed Efficient Multi-Head Attention (EMHA). Extensive experiments show that ESRT achieves competitive results with low computational cost. Compared with the original Transformer which occupies 16,057M GPU memory, ESRT only occupies 4,191M GPU memory. All codes are available at https://github.com/luissen/ESRT.

1. Introduction

Single image super-resolution (SISR) aims at recovering a super-resolution (SR) image from its degraded low-resolution (LR) counterpart, which is a useful technology to overcome resolution limitations in many applications. However, it still is an ill-posed problem since there exist infinite HR images. To address this issue, numerous deep neural networks have been proposed [10, 13, 18, 21, 22, 26, 39, 40, 45]. Although these methods have achieved outstanding performance, they cannot be easily utilized in real applications due to high computation cost and memory storage. To solve this problem, many recurrent networks and lightweight networks have been proposed, such as DRCN [19], SRRFN [23], IMDN [16], IDN [17], CARN [2],ASSLN [46], MAFFSRN [31], and RFDN [27]. All these models concentrate on constructing a more efficient network structure, but the reduced network capacity will lead to poor performance.

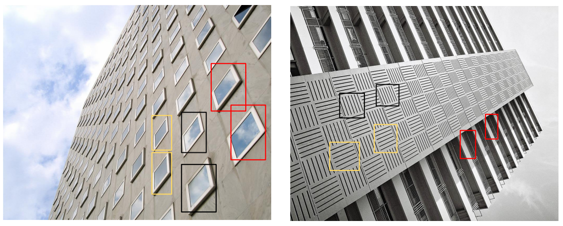
Figure 1. Examples of similar patches in images. These similar patches can help restore details from each other.

As Figure 1 shows, the inner areas of the boxes with the same color are similar to each other. Therefore, these similar image patches can be used as reference images for each other, so that the texture details of the certain patch can be restored with reference patches. Inspired by this, we introduce the Transformer into the SISR task since it has a strong feature expression ability to model such a long-term dependency in the image. In other words, we aim to explore the feasibility of using Transformer in the lightweight SISR task. In recent years, some Vision-Transformer [11, 28] have been proposed for computer vision tasks. However, these methods often occupy heavy GPU memory, which greatly limits their flexibility and application scenarios. Moreover, these methods cannot be directly transferred to SISR since the image restoration task often take a larger resolution image as input, which will take up huge memory.

To solve the aforementioned problems, an Efficient Super-Resolution Transformer (ESRT) is proposed to enhance the ability of SISR networks to capture the long-distance context-dependence while significantly decreasing the GPU memory cost. It is worth noting that ESRT is a hy-

*Corresponding author †Co-first authors

brid architecture, which uses a "CNN+Transformer" pattern to handle the small SR dataset. Specifically, ESRT can be divided into two parts: Lightweight CNN Backbone (LCB) and Lightweight Transformer Backbone (LTB). For LCB, we consider more on reducing the shape of the feature map in the middle layers and maintaining a deep network depth to ensure large network capacity. Inspired by the high-pass filter, we design a High-frequency Filtering Module (HFM) to capture the texture details of the image. With the aid of HFM, a High Preserving Block (HPB) is proposed to extract the potential features efficiently by size variation. For feature extraction, a powerful Adaptive Residual Feature Block (ARFB) is proposed as the basic feature extraction unit with the ability to adaptively adjust the weight of the residual path and identity path. In LTB, an efficient Transformer (ET) is proposed, which use the specially designed Efficient Multi-Head Attention (EMHA) mechanism to decrease the GPU memory consumption. It is worth noting that EMHA just considers the relationship between image blocks in a local region since the pixel in SR image is commonly related to its neighbor pixels. Even though it is a local region, it is much wider than a regular convolution and can extract more useful context information. Therefore, ESRT can learn the relationship between similar local blocks efficiently, making the super-resolved region have more references. The main contributions are as follows

- We propose a Lightweight CNN Backbone (LCB), which use High Preserving Blocks (HPBs) to dynamically adjust the size of the feature map to extract deep features with a low computational cost.
- We propose a Lightweight Transformer Backbone (LTB) to capture long-term dependencies between similar patches in an image with the help of the specially designed Efficient Transformer (ET) and Efficient Multi-Head Attention (EMHA) mechanism.
- A novel model called Efficient SR Transformer (ESRT) is proposed to effectively enhance the feature expression ability and the long-term dependence of similar patches in an image, so as to achieve better performance with low computational cost.

2. Related Works

2.1. CNN-based SISR Models

Recently, many CNN-base models have been proposed for SISR. For example, SRCNN [10] first introduces the deep CNN into SISR and achieves promising results. EDSR [26] optimizes the residual block by removing unnecessary operations and expanding the model size. RCAN [44] proposes a deep residual network with residual-in-residual architecture and channel attention mechanism. SAN [9] presents a second-order attention network to enhance the feature expression and feature correlation learning. IDN [17] compresses the model size by using the group convolution and combining short-term and long-term features. IMDN [16] improves the architecture of IDN and introduces the information multi-distillation blocks to extract the hierarchical features effectively. LatticeNet [29] designs the lattice block that simulates the realization of Fast Fourier Transformation with the butterfly structure. Although these models achieved competitive results, they are pure CNN-based model. This means that they can only extract local features and cannot learn the global information, which is not conducive to the restoration of texture details.

2.2. Vision Transformer

The breakthroughs of Transformer in NLP has leaded to a great interest in the computer vision community. The key idea of Transformer is "self-attention", which can capture long-term information between sequence elements. By adapting Transformer in vision tasks, it has been successfully applied in image recognition [11,24,36], object detection [7,47], and low-level image processing [8,41]. Among them, ViT [11] is the first work to replace the standard convolution with Transformer. To produce the sequence elements, ViT flattened the 2D image patches in a vector and fed them into the Transformer. IPT [8] used a novel Transformer-based network as the pre-trained model for low-level image restoration tasks. SwinIR [25] introduced Swin Transformer [28] into SISR and show the great promise of Transformer in SISR. Although these methods achieved promising results, they require a lot of training data and need heavy GPU memory to train the model, which is not suitable for practical applications. Hence, we aim to explore a more efficient vision-Transformer for SISR.

3. Efficient Super-Resolution Transformer

As shown in Figure 2, Efficient Super-Resolution Transformer (ESRT) mainly consists of four parts: shallow feature extraction, Lightweight CNN Backbone (LCB), Lightweight Transformer Backbone (LTB), and image reconstruction. Define I_{LR} and I_{SR} as the input and output of ESRT, respectively. Firstly, we extract the shallow feature from I_{LR} with a convolutional layer

$$F_0 = f_s(I_{LR}), \tag{1}$$

where f_s denotes the shallow feature extraction layer. F_0 is the extracted shallow feature, which is then used as the input of LCB with several High Preserving Blocks (HPBs)

$$F_n = \zeta^n(\zeta^{n-1}(...(\zeta^1(F_0)))), \tag{2}$$

where ζ^n denotes the mapping of n-th HPB and F_n represents the output of n-th HPB. All outputs of HPB are con-

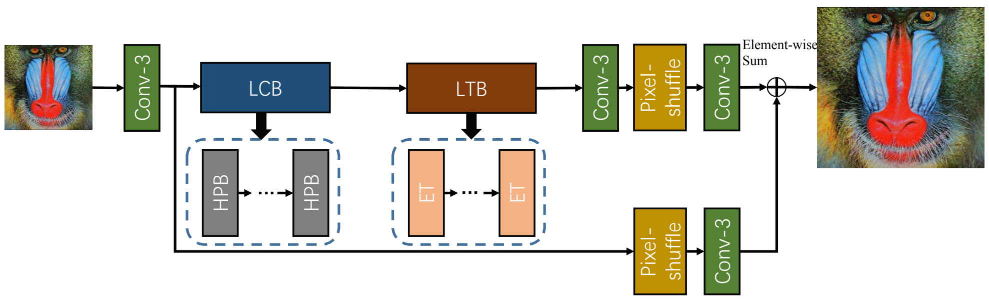

Figure 2. The architecture of the proposed Efficient Super-Resolution Transformer. Among them, LCB, LTB, HPB, and ET stand for the Lightweight CNN Backbone, the Lightweight Transformer Backbone, high preserving block, and efficient Transformers, respectively.

catenated to be sent to LTB with several Efficient Transformers (ETs) to fuse these intermediate features

$$F_d = \phi^n(\phi^{n-1}(...(\phi^1([F_1, F_2, ..., F_n])))), \tag{3}$$

where F_d is the output of LTB and ϕ stands for the operation of ET. Finally, F_d and F_0 are simultaneously fed into the reconstruction module to get the SR image I_{SR}

$$I_{SR} = f(f_p(f(F_d))) + f(f_p(F_0)), \tag{4}$$

where f and f_p stand for the convolutional layer and Pixel-Shuffle layer, respectively.

3.1. Lightweight CNN Backbone (LCB)

The role of Lightweight CNN Backbone (LCB) is to extract potential SR features in advance, so that the model has the initial ability of super-resolution. According to Figure 2, we can observe that LCB is mainly composed of a series of High Preserving Blocks (HPBs).

High Preserving Block (HPB). Previous SR networks usually keep the spatial resolution of feature maps unchanged during processing. In this work, in order to reduce the computational cost, a novel High Preserving Block (HPB) is proposed to reduce the resolution of processing features. However, the reduction of the size of feature maps always leads to the loss of image details, which causes visually unnatural SR images. To solve this problem, in HPB, we creatively preserve the High-frequency Filtering Module (HFM) and Adaptive Residual Feature Block (ARFB).

As shown in Figure 3, an ARFB is first use to extract F_{n-1} as the input features for HFM. Then, HFM is used to calculate the high-frequency information (marked as P_{high}) of the features. After the P_{high} is obtained, we reduce the size of the feature map to reduce computational cost and feature redundancy. The downsampled feature maps are denoted as $F^{'}_{n-1}$. For $F^{'}_{n-1}$, several ARFBs are utilized to explore the potential information for completing the SR image. It is worth noting that these ARFBs share weights to reduce parameters. Meanwhile, a single ARFB is used to

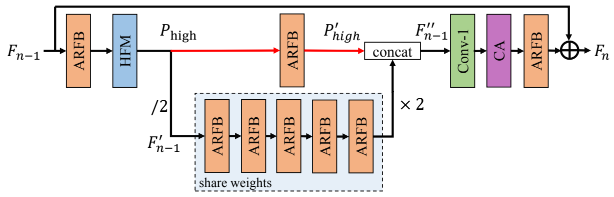

Figure 3. The architecture of the proposed High Preserving Block (HPB), which mainly consists of High-frequency Filtering Module (HFM) and Adaptive Residual Feature Blocks (ARFBs).

process the P_{high} to align the feature space with $F^{'}_{n-1}$. After feature extraction, $F^{'}_{n-1}$ is upsampled to the original size by bilinear interpolation. After that, we fuse the $F^{'}_{n-1}$ with $P^{'}_{high}$ for preserving the initial details and obtain the feature $F^{''}_{n-1}$. This operation can be expressed as

$$F^{''}_{n-1} = [f_a(P_{high}), \uparrow f_a^{\circlearrowright^5}(\downarrow F^{'}_{n-1})], \tag{5}$$

where $\uparrow$ and $\downarrow$ denote the upsampling and downsampling operations, respectively. f_a denotes the operation of ARFB. To achieve good trade-off between the model size and performance, we use five ARFBs in this part according to ablation studies and define it as $f_a^{\circlearrowright^5}$.

For $F^{''}_{n-1}$, as it is concatenated by two features, a 1×1 convolution layer is used to reduce the channel number. Then, a channel attention module [14] is employed to highlight channels with high activated values. Finally, an ARFB is used to extract the final features and the global residual connection is proposed to add the original features F_{n-1} to F_n. The goal of this operation is to learn the residual information from the input and stabilize the training.

3.2. High-frequency Filtering Module (HFM)

Since the Fourier Transform is difficult to embed in CNN, a differentiable HFM is proposed in this work. The target of HFM is to estimate the high-frequency information of the image from the LR space. As shown in Figure 4, assuming the size of the input feature map T_L is $C \times H \times W$,

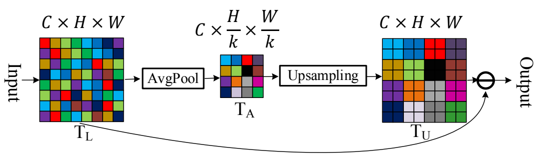

Figure 4. The schematic diagram of the proposed HFM module.

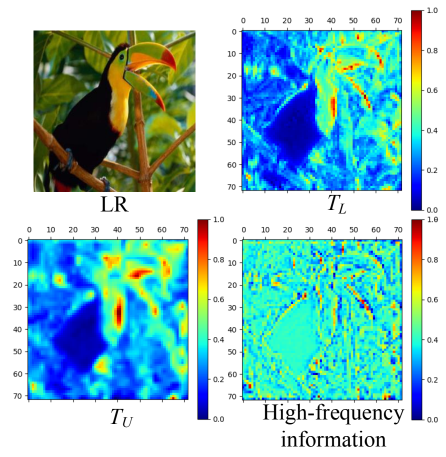

Figure 5. Visual activation maps of T_L, T_U, and obtained high-frequency information. Best viewed in color.

an average pooling layer is first applied to T_L:

$$T_A = avgpool(T_L, k), \tag{6}$$

where k denotes the kernel size of the pooling layer and the size of the intermediate feature map T_A is $C \times \frac{H}{k} \times \frac{W}{k}$. Each value in T_A can be viewed as the average intensity of each specified small area of T_L. After that, T_A is upsampled to get a new tensor T_U of size $C \times H \times W$. T_U is regarded as an expression of the average smoothness information compared with the original T_L. Finally, T_U is element-wise subtracted from T_L to obtain the high-frequency information.

The visual activation maps of T_L, T_U, and high-frequency information are also shown in Figure 5. It can be observed that the T_U is more smooth than T_L as it is the average information of the T_L. Meanwhile, the high-frequency information retains the details and edges of the feature map before downsampling. Therefore, it is essential to save these information.

3.2.1 Adaptive Residual Feature Block (ARFB)

As explored in ResNet [12] and VDSR [18], when the depth of the model grows, the residual architecture can mitigate the gradient vanishing problem and augment the representation capacity of the model. Inspired by them, a Adaptive Residual Feature Block (ARFB) is proposed as the basic feature extraction block. As shown in Figure 6, ARFB contains two Residual Units (RUs) and two convolutional layers. To save memory and the number of parameters, RU is made up of two modules: Reduction and Expansion. For Reduction, the channels of the feature map are reduced by half and recovered in Expansion. Meanwhile, a residual scaling with adaptive weights (RSA) is designed to dynamically adjust the importance of residual path and identity path. Compared with fixed residual scaling, RSA can improve the flow of gradients and automatically adjust the content of the residual feature maps for the input feature map. Assume that x_{ru} is the input of RU, the process of RU can be formulated as:

$$y_{ru} = \lambda_{res} \cdot f_{ex}(f_{re}(x_{ru})) + \lambda_x \cdot x, \tag{7}$$

where y_{ru} is the output of RU, f_{re} and f_{ex} represent the Reduction and Expansion operations, λ_{res} and λ_x are two adaptive weights for two paths, respectively. These two operations use 1×1 convolutional layers to change the number of channels to achieve the functions of reduction and expansion. Meanwhile, the outputs of two RUs are concatenated followed by a 1×1 convolutional layer to fully utilize the hierarchical features. In the end, a 3×3 convolutional layer is adopted to reduce the channels of the feature map and extract valid information from the fused features.

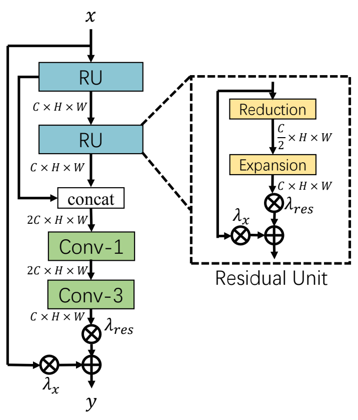

Figure 6. The complete architecture of the proposed ARFB.

3.3. Lightweight Transformer Backbone (LTB)

In SISR, similar image blocks within the image can be used as reference images to each other, so that the texture details of the current image block can be restored with reference to other image blocks, which is proper to use Transformer. However, previous variants of vision Transformer commonly need heavy GPU memory cost, which hinders the development of Transformer in the vision area. In this paper, we propose a Lightweight Transformer Backbone (LTB). LTB is composed of specially designed Efficient Transformers (ETs), which can capture the long-term

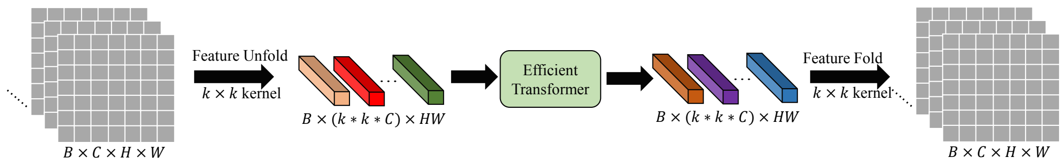
Feature Unfold
$k \times k$ kernel
$B \times C \times H \times W$
$B \times (k * k * C) \times HW$
Efficient Transformer
$B \times (k * k * C) \times HW$
Feature Fold
$k \times k$ kernel
$B \times C \times H \times W$

Figure 7. The complete pre- and post-processing for the Efficient Transformer (ET). Specifically, we use the unfolding technique to split the feature maps into patches and use the fold operation to reconstruct the feature map.

dependence of similar local regions in the image at a low computational cost.

3.3.1 Pre- and Post-processing for ET

The standard Transformer takes a 1-D sequence as input, learning the long-distance dependency of the sequence. However, for the vision task, the input is always a 2-D image. The common way to turn a 2-D image into a 1-D sequence is to sort the pixels in the image one by one. However, this method will lose the unique local correlation of the image, leading to sub-optimal performance. In ViT [11], the 1-D sequence is generated by non-overlapping block partitioning, which means there is no pixel overlap between each block. According to our experiments, these pre-processing methods are not suitable for SISR. Therefore, a novel processing way is proposed to handle the feature maps.

As shown in Figure 7, we use the unfolding technique to split the feature maps into patches and each patch is considered as a "word". Specifically, the feature maps $F_{ori} \in \mathbb{R}^{C \times H \times W}$ are unfolded (by $k \times k$ kernel) into a sequence of patches, i.e., $F_{p_i} \in \mathbb{R}^{k^2 \times C}$, $i = \{1, ..., N\}$, where $N = H \times W$ is the amount of patches. Here, the learnable position embeddings are eliminated for each patch since the "Unfold" operation automatically reflects the position information for each patch. After that, those patches F_p are directly sent to the ET. The output of ET has the same shape as the input and we use the "Fold" operation to reconstruct feature maps.

3.3.2 Efficient Transformer (ET)

For simplicity and efficiency, like ViT [11], ET only uses the encoder structure of the standard Transformer. As shown in Figure 8, in the encoder of ET, there consists of an Efficient Multi-Head Attention (EMHA) and an MLP. Meanwhile, layer-normalization [4] is employed before every block, and the residual connection is also applied after each block. Assume the input embeddings are E_i, the output embeddings E_o can be obtained by

$$\begin{aligned} E_{m1} &= EMHA(Norm(E_i)) + E_i, \\ E_o &= MLP(Norm(E_{m1})) + E_{m1}, \end{aligned} \tag{8}$$

where E_o is the output of the ET, $EMHA(\cdot)$ and $MLP(\cdot)$ represent the EMHA and MLP operations, respectively.

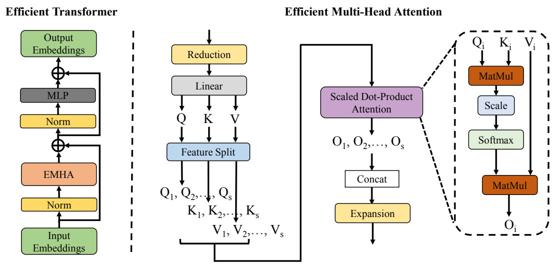

Figure 8. Architecture of Efficient Transformer. EMHA is the Efficient Multi-Head Attention. MatMul is the matrix multiplication.

Efficient Multi-Head Attention (EMHA). As shown in Figure 8, there are several modifications in EMHA to make EMHA more efficient and occupy lower GPU memory cost compared with the original MHA [37]. Assume the shape of the input embedding E_i is $B \times C \times N$. Firstly, a Reduction layer is used to reduce the number of channels by half ($B \times C_1 \times N, C_1 = \frac{C}{2}$). After that, a linear layer is adopted to project the feature map into three elements: Q (query), K (keys), and V (values). As employed in Transformer, we linearly project the Q, K, V, and m times to perform the multi-head attention. m is the number of heads. Next, the shape of three elements is reshaped and permuted to $B \times m \times N \times \frac{C_1}{m}$. In original MHA, Q, K, V are directly used to calculate the self-attention with large-scale matrix multiplication, which cost huge memory. Assume Q and K calculate the self-attention matrix with shape $B \times m \times N \times N$. Then this matrix computes the self-attention with V, the dimension in 3-th and 4-th are $N \times N$. For SISR, the images usually have high resolution, causing that N is very large and the calculation of self-attention matrix consumes a lot of GPU memory cost and computational cost. To address this issue, a Feature Split (FS) Module is used to split Q, K, and V into s equal segments with splitting factor s since the predicted pixels in super-resolved images often only depend on the local adjacent areas in LR. Therefore, the dimension in 3-th and 4-th of the last self-matrix is $\frac{N}{s} \times \frac{N}{s}$, which can significantly reduce the computational and GPU memory cost. Denote these segments as $Q_1, ..., Q_s$, $K_1, ..., K_s$, and $V_1, ..., V_s$. Each triplet of these segments is applied with a Scaled Dot-Product Attention (SDPA) operation, respectively. The structure of SDPA is also shown in Figure 8, which just omits the $Mask$ oper-

Method	Scale	Params	Set5	Set14	BSD100	Urban100	Manga109
			PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
VDSR [18]	×3	666K	33.66 / 0.9213	29.77 / 0.8314	28.82 / 0.7976	27.14 / 0.8279	32.01 / 0.9340
MemNet [34]		678K	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	27.56 / 0.8376	32.51 / 0.9369
EDSR-baseline [26]		1,555K	34.37 / 0.9270	30.28 / 0.8417	29.09 / 0.8052	28.15 / 0.8527	33.45 / 0.9439
SRMDNF [43]		1,528K	34.12 / 0.9254	30.04 / 0.8382	28.97 / 0.8025	27.57 / 0.8398	33.00 / 0.9403
CARN [2]		1,592K	34.29 / 0.9255	30.29 / 0.8407	29.06 / 0.8034	28.06 / 0.8493	33.50 / 0.9440
IMDN [16]		703K	34.36 / 0.9270	30.32 / 0.8417	29.09 / 0.8046	28.17 / 0.8519	33.61 / 0.9445
RFDN-L [27]		633K	34.47 / 0.9280	30.35 / 0.8421	29.11 / 0.8053	28.32 / 0.8547	33.78 / 0.9458
MAFFSRN [31]		807K	34.45 / 0.9277	30.40 / 0.8432	29.13 / 0.8061	28.26 / 0.8552	- / -
LatticeNet [29]		765K	**34.53 / 0.9281**	30.39 / 0.8424	**29.15** / 0.8059	28.33 / 0.8538	- / -
ESRT(ours)		770K	34.42 / 0.9268	**30.43 / 0.8433**	**29.15 / 0.8063**	**28.46 / 0.8574**	**33.95 / 0.9455**
VDSR [18]	×4	666K	31.35 / 0.8838	28.01 / 0.7674	27.29 / 0.7251	25.18 / 0.7524	28.83 / 0.8870
MemNet [34]		678K	31.74 / 0.8893	28.26 / 0.7723	27.40 / 0.7281	25.50 / 0.7630	29.42 / 0.8942
EDSR-baseline [26]		1,518K	32.09 / 0.8938	28.58 / 0.7813	27.57 / 0.7357	26.04 / 0.7849	30.35 / 0.9067
SRMDNF [43]		1,552K	31.96 / 0.8925	28.35 / 0.7787	27.49 / 0.7337	25.68 / 0.7731	30.09 / 0.9024
CARN [2]		1,592K	32.13 / 0.8937	28.60 / 0.7806	27.58 / 0.7349	26.07 / 0.7837	30.47 / 0.9084
IMDN [16]		715K	32.21 / 0.8948	28.58 / 0.7811	27.56 / 0.7353	26.04 / 0.7838	30.45 / 0.9075
RFDN-L [27]		643K	32.28 / 0.8957	28.61 / 0.7818	27.58 / 0.7363	26.20 / 0.7883	30.61 / 0.9096
MAFFSRN [31]		830K	32.20 / 0.8953	26.62 / 0.7822	27.59 / 0.7370	26.16 / 0.7887	- / -
LatticeNet [29]		777K	**32.30 / 0.8962**	28.68 / 0.7830	27.62 / 0.7367	26.25 / 0.7873	- / -
ESRT (ours)		751K	32.19 / 0.8947	**28.69 / 0.7833**	**27.69 / 0.7379**	**26.39 / 0.7962**	**30.75 / 0.9100**

Table 1. Quantitative comparison with SISR models. The Best and the second-best results are **highlighted** and underlined, respectively.

ation. Afterward, all the outputs $(O_1, O_2, ..., O_s)$ of SDPA are concatenated together to generate the whole output feature O. Finally, an Expansion layer is used to recover the number of channels.

4. Experiments

4.1. Datasets and Metrics

In this work, we use DIV2K [35] as the training dataset. For evaluation, we use five benchmark datasets, including Set5 [5], Set14 [42], BSD100 [30], Urban100 [15], and Manga109 [3]. Meanwhile, PSNR and SSIM are used to evaluate the performance of the reconstructed SR images.

4.2. Implementation Details

Training Setting. During training, we randomly crop 16 LR image patches with the size of 48×48 as inputs in each epoch. Random horizontal flipping and 90 degree rotation is used for data augment. The initial learning rate is set to 2×10^{-4} and decreased half for every 200 epochs. The model is trained by Adam optimizer with a momentum equal to 0.9. Meanwhile, $L1$ loss is used as it can produce more sharp images compared to $L2$ loss. Meanwhile, we implement ESRT with the PyTorch framework and train ESRT roughly takes two days with one GTX1080Ti GPU.

Implements Details. In ESRT, we set 3×3 as the size of all convolutional layers except for the Reduction module, whose kernel size is 1×1. Each convolutional layer has 32 channels except for the fusion layer which is twice. For the image reconstruction part, following most previous methods, we use PixelShuffle [32] to upscale the last coarse features to fine features. The k in HFP is set to 2, which means that the feature map is down-scaled by half. Meanwhile, the number of HPB is set to 3, the initial value of learnable weight in ARFB is set to 1, and the number of ET in LTB is set to 1 to save the GPU memory. In addition, the splitting factor s in ET is set to 4, the k in pre- and post-process of ET is set to 3, and the head number m in EMHA is set to 8, respectively.

4.3. Comparisons with Advanced SISR Models

In TABLE 1, we compare ESRT with other advanced SISR models. Obviously, our ESRT achieves competitive results under all scaling factors. It can be seen that although the performance of EDSR-baseline is close to ESRT, it has almost twice as many parameters as ESRT. Meanwhile, the number of MAFFSRN and LatticeNet is close to ESRT, but ESRT achieves better results than them. Moreover, we can observe that our ESRT performs much better than other models on Urban100. This is because there are many similar patches in each image of this dataset. Therefore, the introduced LTB in our ESRT can used to capture the long-term dependencies among these similar image patches and learn their relevance, thus achieve better results.

In Figure 9, we also provide the visual comparison between ESRT and other lightweight SISR models on $\times 2$, $\times 3$, and $\times 4$. Obviously, SR images reconstructed by our ESRT contains more accurate texture details, especially in the edges and lines. It is worth noting that in the $\times 4$ scale, the gap between ESRT and other SR models is more apparent. This benefits from the effectiveness of the proposed Efficient Transformer, which can learn more information from other clear areas. All these experiments validate the effectiveness of the proposed ESRT.

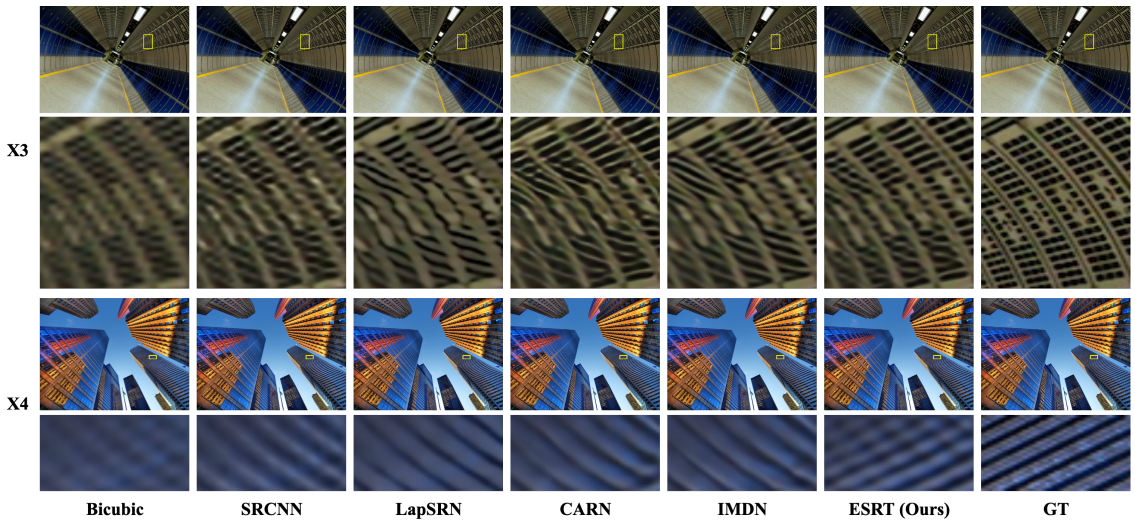

Figure 9. Visual comparison with lightweight SISR models. Obviously, ESRT can reconstruct realistic SR images with sharper edges.

Method	Layers	RL	Param.	FLOPs (x4)	Running time
VDSR [18]	20	Yes	0.67M	612.6G	0.00597s
LapSRN [20]	27	Yes	0.25M	149.4G	0.00330s
DRRN [33]	52	No	0.30M	6796.9G	0.08387s
CARN [2]	34	Yes	1.6M	90.9G	0.00278s
IMDN [16]	34	Yes	0.7M	40.9G	0.00258s
ESRT	163	Yes	0.68M	67.7G	0.01085s

Table 2. Network structure settings comparison between our ESRT and other lightweight SISR models (the input size is 1280 $\times$ 720).

4.4. Comparison on Computational Cost

In TABLE 2, we provide a more detailed comparison of each model. It can be seen that ESRT can achieve 163 layers while still achieves the second-least FLOPs (67.7G) among these methods. This is benefited from the proposed HPB and ARFB, which can efficiently extract useful features and preserving the high-frequency information. Meanwhile, we can observe that the execution time is short even though ESRT uses the Transformer architecture. The increased time is completely acceptable compared to CARN and IMDN. In addition, we also visualize the trade-off analysis between the number of model parameters and performance in Figure 10. Obviously, we can see that our ERST achieves a good trade-off between the size and performance of the model.

4.5. Network Investigations

4.5.1 Study of High Preserving Block (HPB)

HPB is an important component of ESRT, which not only can reduce the model size but maintain the high performance of the model. To prove it, we explore the effectiveness of each component of ESRT in TABLE 3. According to

Case Index	1	2	3	4
HFM		√	√	√
CA	√	√		√
ARFB	√	√	√	
RB				√
Parameters	658K	751K	724K	972K
PSNR	32.02dB	<u>32.19dB</u>	32.08dB	**32.20dB**

Table 3. Study of each component in HPB on Set5 ($\times$4).

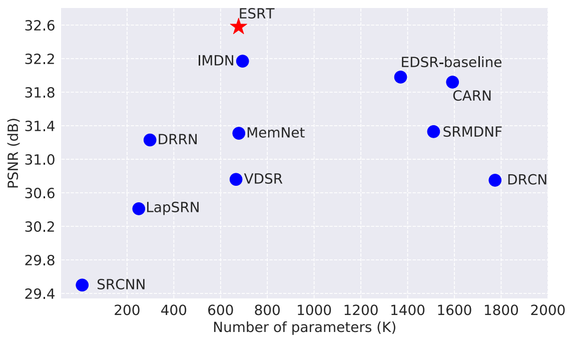

Figure 10. Study the trade-off between the number of model parameters and performance on Urban100 ($\times$2).

Cases 1, 2, and 3, we can observe that the introduced HFM and CA can effectively improve the model performance at the cost of a few parameters. According to Cases 2 and 4, we can see that if RB is used to represent ARFB, the PSNR result just rises 0.01dB but the number of parameters go up to 972K. This means that ARFB can significantly reduce the model parameters while maintaining excellent performance. All these results fully illustrate the necessity and effectiveness of these modules and mechanisms in HPB.

Case	PSNR(dB)	Parame.(K)	GPU Memory
w/o TR	31.96	554	1931M
Original TR [38]	32.14	971	16057M
1 ET	**32.18**	751	4191M
2 ET	32.25	949	6499M

Table 4. Study of Efficient Transformer (ET) on Set5 (×4). The GPU memory here refers to the cost of the model during training, which patch_size = 48*48 and batch_size=16.

Scale	Model	Param	Set5	Set14	Urban100
×3	RCAN [44]	16M	**34.74dB**	**30.65dB**	29.09dB
	RCAN/2+ET	8.7M	34.69dB	30.63dB	**29.16dB**
×4	RCAN [44]	16M	**32.63dB**	28.87dB	26.82dB
	RCAN/2+ET	8.7M	32.60dB	**28.90dB**	**26.87dB**

Table 5. Comparison between RCAN and RCAN/2+ET.

4.5.2 Study of Efficient Transformer (ET)

To capture the long-term dependencies of similar local regions in the image, we introduced the Transformer and proposed a Efficient Transformer (ET). In TABLE 4, we analyze the model with and without Transformer. It can be see that if ESRT removes the Transformer, the model performance descends obviously from 32.18dB to 31.96dB. This is because the introduced Transformer can make full advantage of the relationship between similar image patches in an image. In addition, we compare our ET with the original Transformer [11] in the table. Our model (1ET) achieves better results with fewer parameters and GPU memory consumption (1/4). This experiment fully verified the effectiveness of the proposed ET. Meanwhile, we can also see that when the number of ET increases, the model performance will be further improved. However, it is worth noting that the model parameters and GPU memory will also increase when the number of ET increases. Therefore, to achieve a good balance between the size and performance of the model, only one ET is used in the final ESRT.

In order to verify the effectiveness and universality of the proposed ET, we also introduce ET into RCAN [44]. It is worth noting that we use a small version of RCAN (the residual group number is set to 5) and add the ET before the reconstruction part. According to TABLE 5, we can see that the performance of the model "RCAN/2+ET" is close or even better than the original RCAN with fewer parameters. This further proves the effectiveness and universality of ET, which can be easily transplanted to any existing SISR models to further improve the performance of model.

4.6. Real Image Super-Resolution

To further verify the validity of the model, we also compare our ESRT with some classic lightweight SR models on the real image dataset (RealSR [6]). According to TABLE 6, we can observe that ESRT achieves better results than IMDN. In addition, ESRT achieves better performance than LK-KPN on ×4, which was specifically designed for the real SR task. This experiment further verifies its effectiveness on real images.

Scale	IMDN [16]		LK-KPN [6]		ESRT (Ours)	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
×3	30.29	0.857	**30.60**	**0.863**	30.38	0.857
×4	28.68	0.815	28.65	**0.820**	**28.78**	0.815

Table 6. Comparison with advanced SISR methods on RealSR.

Method	Parame.	GPU Memory	BSD100	Manga109
SwinIR	886K	6966M	29.20/0.8082	33.98/0.9478
ESRT	770K	4191M	29.15/0.8063	33.95/0.9455

Table 7. A detailed comparison of SwinIR and ESRT (×4).

4.7. Comparison with SwinIR

The EMHA in our ESRT is similar to the Swin Transformer layer of SwinIR [25]. However, SwinIR uses a sliding window to solve the high computation problem of the Transformer while ESRT uses a splitting factor to reduce the GPU memory consumption. According to Table 7, compared with SwinIR, ESRT achieves close performance with fewer parameters and GPU memory. It is worth noting that SwinIR uses an extra dataset (Flickr2K [1]) for training, which is the key to further improving the model performance. For a fair comparison with methods such as IMDN, we did not use this external dataset in this work.

5. Conclusion

In this work, we proposed a novel Efficient Super-Resolution Transformer (ESRT) for SISR. ESRT first utilizes a Lightweight CNN Backbone (LCB) to extract deep features and then uses a Lightweight Transformer Backbone (LTB) to model the long-term dependence between similar local regions in an image. In LCB, we proposed a High Preserving Block (HPB) to reduce the computational cost and retain high-frequency information with the help of the specially designed High-frequency Filtering Module (HFM) and Adaptive Residual Feature Block (ARFB). In LTB, an Efficient Transformer (ET) is designed to enhance the feature representation ability with lower GPU memory occupation under the help of the proposed Efficient Multi-head Attention (EMHA). Extensive experiments demonstrate that ESRT achieves the best trade-off between model performance and computation cost.

6. Acknowledgment

This work was supported by National Key R&D Program of China (2021YFE0203700), CRF (8730063), National Natural Science Foundation of China (U1613209), and Science and Technology Plan Projects of Shenzhen (JCYJ20190808182209321, JCYJ20200109140410340).

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017. 8

[2] N. Ahn, B. Kang, and K. A. Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, pages 252–268, 2018. 1, 6, 7

[3] Kiyoharu Aizawa, Azuma Fujimoto, Atsushi Otsubo, Toru Ogawa, Yusuke Matsui, Koki Tsubota, and Hikaru Ikuta. Building a manga dataset "manga109" with annotations for multimedia applications. *IEEE Multimedia*, 27(2):8–18, 2020. 6

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[5] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. 2012. 6

[6] Jianrui Cai, Hui Zeng, Hongwei Yong, Zisheng Cao, and Lei Zhang. Toward real-world single image super-resolution: A new benchmark and a new model. In *ICCV*, pages 3086–3095, 2019. 8

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020. 2

[8] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *CVPR*, pages 12299–12310, 2021. 2

[9] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *CVPR*, pages 11065–11074, 2019. 2

[10] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE TPAMI*, 38(2):295–307, 2015. 1, 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 5, 8

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 4

[13] Zewei He, Siliang Tang, Jiangxin Yang, Yanlong Cao, Michael Ying Yang, and Yanpeng Cao. Cascaded deep networks with multiple receptive fields for infrared image super-resolution. *IEEE TCSVT*, 29(8):2310–2322, 2018. 1

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018. 3

[15] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015. 6

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

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

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016. 1, 4, 6, 7

[19] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, pages 1637–1645, 2016. 1

[20] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks. *IEEE TPAMI*, 41(11):2599–2613, 2018. 7

[21] Juncheng Li, Faming Fang, Kangfu Mei, and Guixu Zhang. Multi-scale residual network for image super-resolution. In *ECCV*, pages 517–532, 2018. 1

[22] Juncheng Li, Zehua Pei, and Tieyong Zeng. From beginner to master: A survey for deep learning-based single-image super-resolution. *arXiv preprint arXiv:2109.14335*, 2021. 1

[23] Juncheng Li, Yiting Yuan, Kangfu Mei, and Faming Fang. Lightweight and accurate recursive fractal network for image super-resolution. In *ICCVW*, 2019. 1

[24] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021. 2

[25] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1833–1844, 2021. 2, 8

[26] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, pages 136–144, 2017. 1, 2, 6

[27] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *ECCV*, pages 41–55, 2020. 1, 6

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1, 2

[29] Xiaotong Luo, Yuan Xie, Yulun Zhang, Yanyun Qu, Cuihua Li, and Yun Fu. Latticenet: Towards lightweight image super-resolution with lattice block. In *ECCV*, pages 272–289, 2020. 2, 6

[30] David Martin, Charless 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 *ICCV*, pages 416–423, 2001. 6

[31] Abdul Muqeet, Jiwon Hwang, Subin Yang, JungHeum Kang, Yongwoo Kim, and Sung-Ho Bae. Multi-attention based ultra lightweight image super-resolution. In *ECCV*, pages 103–118, 2020. 1, 6

[32] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, 2016. 6

[33] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *CVPR*, pages 3147–3155, 2017. 7

[34] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, pages 4539–4547, 2017. 6

[35] Radu Timofte, Shuhang Gu, Jiqing Wu, and Luc Van Gool. Ntire 2018 challenge on single image super-resolution: Methods and results. In *CVPRW*, pages 852–863, 2018. 6

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021. 2

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. 5

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 8

[39] Yifan Wang, Lijun Wang, Hongyu Wang, and Peihua Li. Resolution-aware network for image super-resolution. *IEEE TCSVT*, 29(5):1259–1269, 2018. 1

[40] Yifan Wang, Lijun Wang, Hongyu Wang, and Peihua Li. Resolution-aware network for image super-resolution. *IEEE TCSVT*, 29:1259–1269, 2019. 1

[41] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *CVPR*, pages 5791–5800, 2020. 2

[42] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE TIP*, 19(11):2861–2873, 2010. 6

[43] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR*, pages 3262–3271, 2018. 6

[44] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, pages 286–301, 2018. 2, 8

[45] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, pages 2472–2481, 2018. 1

[46] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. Aligned structured sparsity learning for efficient image super-resolution. *NeurIPS*, 34, 2021. 1

[47] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2