Supplementary Material of 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

1. Overview

In this work, we proposed a novel Efficient Super-Resolution Transformer (ESRT). In this supplementary material, we provide more experiments to further illustrate the effectiveness and advancement of ESRT. All codes available at https://github.com/luissen/ESRT.

1.1. Comparisons with Advanced SISR Models

1.1.1 Objective Evaluation

In TABLE 1, we compare our ESRT with more than 15 advanced SISR models. Most of them achieve the best results at the time with a well-designed lightweight network. Obviously, our ESRT achieves competitive results with a small amount of parameters. It can be seen that our ESRT performs much better than other models on Urban100 and Manga109 datasets. This is because there are many similar patches in each image of these datasets. 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 future improve the performance of the model.

1.1.2 Subjective Evaluation

In Figure 1, we also provide more visual comparison between ESRT and other advanced SISR models. 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. However, we also notice that there are still contains some error lines in our reconstructed images. This is because the LR image is a down-sampled image, some regional features in the image are severely damaged. For these areas, it is difficult for the model to match the available reference patches for its learning. Therefore, the reconstructed lines of these areas are not straight still.

1.2. Network Investigations

1.2.1 Study of Adaptive Residual Feature Block

In this work, we proposed a powerful Adaptive Residual Feature Block (ARFB) for feature extraction. In this part, we provide a detailed ablation study to validate the effectiveness of ARFB. Specifically, HPB is composed of an upper branch and a lower branch. Among them, the upper branch is used to extract high-frequency information and the lower branch is used to mine potential features. It is worth noting that all ARFBs share weights in the lower branch to reduce parameters. This means that this is a recursive component, which helps maximize the use of model parameters. In Table 2, we provide the impact of different ARFBs in the lower branch on model performance. According to the table, we can find: a) The introduced weight sharing strategy can further improve model performance; b) When the number of ARFB is increased, the model performance can be further improved; c) When the number of ARFBs increased to 6, the model performance no longer increased, and even a slight decrease. **Therefore, we use 5 ARFBs in the lower branch on the final HPB to achieve the best results.**

In the upper branch, we only use one ARFB to extract high-frequency information. In Table 3, we provide the impact of different ARFBs in the upper branch on model performance. Obviously, adding more ARFB will further improve model performance. However, it cannot be ignored that as the number of ARFB increases, the number of parameter of the model will also increase, which is not conducive to the construction of lightweight models. Meanwhile, the growth rate of model performance will also slow down. Therefore, we only use one ARFB in HPB to achieve a good balance between model size and performance.

*Corresponding author †Co-first authors

Method	Scale	Params	Set5	Set14	BSD100	Urban100	Manga109
			PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Bicubic		-	33.66 / 0.9299	30.24 / 0.8688	29.56 / 0.8431	26.88 / 0.8403	30.80 / 0.9339
SRCNN [6]		8K	36.66 / 0.9542	32.45 / 0.9067	31.36 / 0.8879	29.50 / 0.8946	35.60 / 0.9663
FSRCNN [7]		13K	37.00 / 0.9558	32.63 / 0.9088	31.53 / 0.8920	29.88 / 0.9020	36.67 / 0.9710
VDSR [11]		666K	37.53 / 0.9587	33.03 / 0.9124	31.90 / 0.8960	30.76 / 0.9140	37.22 / 0.9750
DRCN [12]		1,774K	37.63 / 0.9588	33.04 / 0.9118	31.85 / 0.8942	30.75 / 0.9133	37.55 / 0.9732
LapSRN [13]		251K	37.52 / 0.9591	32.99 / 0.9124	31.80 / 0.8952	30.41 / 0.9103	37.27 / 0.9740
DRRN [20]		298K	37.74 / 0.9591	33.23 / 0.9136	32.05 / 0.8973	31.23 / 0.9188	37.88 / 0.9749
MemNet [21]	$\times 2$	678K	37.78 / 0.9597	33.28 / 0.9142	32.08 / 0.8978	31.31 / 0.9195	37.72 / 0.9740
IDN [10]		553K	37.83 / 0.9600	33.30 / 0.9148	32.08 / 0.8985	31.27 / 0.9196	38.01 / 0.9749
EDSR-baseline [16]		1,370K	37.99 /0.9604	33.57 /0.9175	32.16 / 0.8994	31.98 / 0.9272	38.54 / 0.9769
SRMDNF [24]		1,511K	37.79 / 0.9601	33.32 / 0.9159	32.05 / 0.8985	31.33 / 0.9204	38.07 / 0.9761
CARN [2]		1,592K	37.76 / 0.9590	33.52 / 0.9166	32.09 / 0.8978	31.92 / 0.9256	38.36 / 0.9765
IMDN [9]		694K	38.00 / 0.9605	33.63 / 0.9177	32.19 / 0.8996	32.17 / 0.9283	38.88 / 0.9774
RFDN-L [17]		626K	38.08 / 0.9606	33.67 / 0.9190	32.18 / 0.8996	32.24 / 0.9290	38.95 / 0.9773
MAFFSRN [19]		790K	38.07 / 0.9607	33.59 / 0.9177	32.23 / **0.9005**	32.38 / 0.9308	- / -
LatticeNet [18]		756K	**38.15 / 0.9610**	**33.78 / 0.9193**	**32.25 / 0.9005**	32.43 / 0.9302	- / -
ESRT(ours)		677K	38.03 / 0.9600	33.75 / 0.9184	**32.25** / 0.9001	**32.58 / 0.9318**	**39.12 / 0.9774**
Bicubic		-	30.39 / 0.8682	27.55 / 0.7742	27.21 / 0.7385	24.46 / 0.7349	26.95 / 0.8556
SRCNN [6]		8K	32.75 / 0.9090	29.30 / 0.8215	28.41 / 0.7863	26.24 / 0.7989	30.48 / 0.9117
FSRCNN [7]		13K	33.18 / 0.9140	29.37 / 0.8240	28.53 / 0.7910	26.43 / 0.8080	31.10 / 0.9210
VDSR [11]		666K	33.66 / 0.9213	29.77 / 0.8314	28.82 / 0.7976	27.14 / 0.8279	32.01 / 0.9340
DRCN [12]		1,774K	33.82 / 0.9226	29.76 / 0.8311	28.80 / 0.7963	27.15 / 0.8276	32.24 / 0.9343
LapSRN [13]		502K	33.81 / 0.9220	29.79 / 0.8325	28.82 / 0.7980	27.07 / 0.8275	32.21 / 0.9350
DRRN [20]		298K	34.03 / 0.9244	29.96 / 0.8349	28.95 / 0.8004	27.53 / 0.8378	32.71 / 0.9379
MemNet [21]	$\times 3$	678K	34.09 / 0.9248	30.00 / 0.8350	28.96 / 0.8001	27.56 / 0.8376	32.51 / 0.9369
IDN [10]		553K	34.11 / 0.9253	29.99 / 0.8354	28.95 / 0.8013	27.42 / 0.8359	32.71 / 0.9381
EDSR-baseline [16]		1,555K	34.37 / 0.9270	30.28 / 0.8417	29.09 / 0.8052	28.15 / 0.8527	33.45 / 0.9439
SRMDNF [24]		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 [9]		703K	34.36 / 0.9270	30.32 / 0.8417	29.09 / 0.8046	28.17 / 0.8519	33.61 / 0.9445
RFDN-L [17]		633K	34.47 / 0.9280	30.35 / 0.8421	29.11 / 0.8053	28.32 / 0.8547	33.78 / 0.9458
MAFFSRN [19]		807K	34.45 / 0.9277	30.40 / 0.8432	29.13 / 0.8061	28.26 / 0.8552	- / -
LatticeNet [18]		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**
Bicubic		-	28.42 / 0.8104	26.00 / 0.7027	25.96 / 0.6675	23.14 / 0.6577	24.89 / 0.7866
SRCNN [6]		8K	30.48 / 0.8628	27.50 / 0.7513	26.90 / 0.7101	24.52 / 0.7221	27.58 / 0.8555
FSRCNN [7]		13K	30.72 / 0.8660	27.61 / 0.7550	26.98 / 0.7150	24.62 / 0.7280	27.90 / 0.8610
VDSR [11]		666K	31.35 / 0.8838	28.01 / 0.7674	27.29 / 0.7251	25.18 / 0.7524	28.83 / 0.8870
DRCN [12]		1,774K	31.53 / 0.8854	28.02 / 0.7670	27.23 / 0.7233	25.14 / 0.7510	28.93 / 0.8854
LapSRN [13]		502K	31.54 / 0.8852	28.09 / 0.7700	27.32 / 0.7275	25.21 / 0.7562	29.09 / 0.8900
DRRN [20]		298K	31.68 / 0.8888	28.21 / 0.7720	27.38 / 0.7284	25.44 / 0.7638	29.45 / 0.8946
MemNet [21]	$\times 4$	678K	31.74 / 0.8893	28.26 / 0.7723	27.40 / 0.7281	25.50 / 0.7630	29.42 / 0.8942
IDN [10]		553K	31.82 / 0.8903	28.25 / 0.7730	27.41 / 0.7297	25.41 / 0.7632	29.41 / 0.8942
EDSR-baseline [16]		1,518K	32.09 / 0.8938	28.58 / 0.7813	27.57 / 0.7357	26.04 / 0.7849	30.35 / 0.9067
SRMDNF [24]		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 [9]		715K	32.21 / 0.8948	28.58 / 0.7811	27.56 / 0.7353	26.04 / 0.7838	30.45 / 0.9075
RFDN-L [17]		643K	32.28 / 0.8957	28.61 / 0.7818	27.58 / 0.7363	26.20 / 0.7883	30.61 / 0.9096
MAFFSRN [19]		830K	32.20 / 0.8953	26.62 / 0.7822	27.59 / 0.7370	26.16 / 0.7887	- / -
LatticeNet [18]		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.

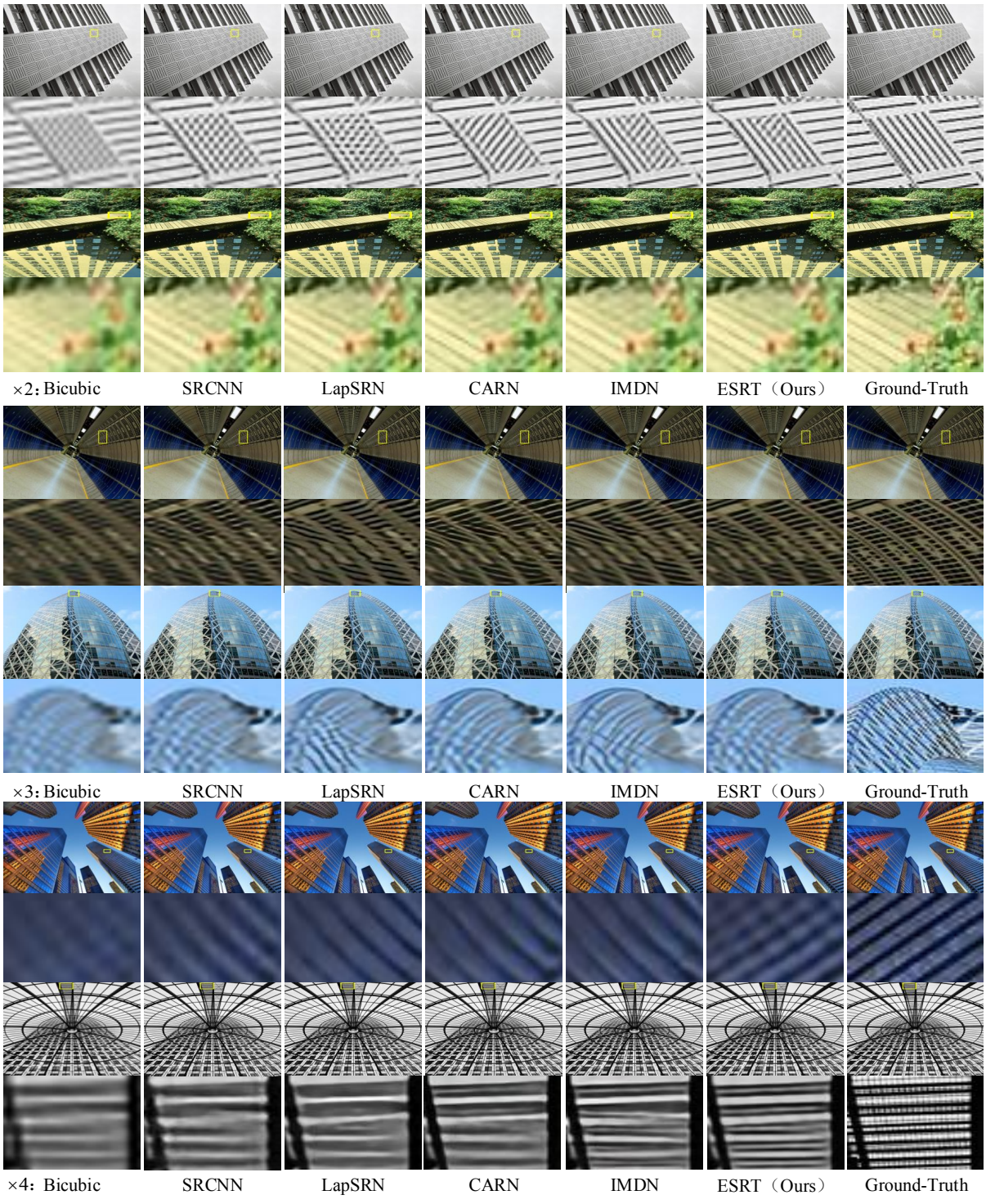

Figure 1. Visual comparison with lightweight SISR models. Obviously, ESRT can reconstruct realistic SR images with sharper edges.

ARFBs	Set5	Set14	BSD100	Urban100	Manga109
1	32.07dB	28.53dB	27.58dB	26.23dB	30.60dB
3	32.11dB	28.60dB	27.63dB	26.38dB	30.70dB
5	**32.19dB**	**28.69dB**	27.69dB	**26.39dB**	**30.75dB**
6	32.18dB	**28.69dB**	**27.70dB**	26.37dB	30.74dB

Table 2. Study on the impact of different numbers of ARFB in the low branch on model performance (x4).

ARFBs	Set5	Set14	BSD100	Urban100	Manga109
0	32.17dB	28.59dB	27.55dB	26.22dB	30.61dB
1	32.19dB	28.69dB	27.69dB	26.39dB	30.75dB
2	**32.20dB**	**28.71dB**	27.71dB	**26.44dB**	**30.78dB**

Table 3. Study on the impact of different numbers of ARFB in the upper branch on model performance (x4).

Adaptive	Set5	Set14	BSD100	Urban100	Manga109
X	32.13dB	28.58dB	27.56dB	26.24dB	30.62dB
$\surd$	**32.19dB**	**28.69dB**	**27.69dB**	**26.39dB**	**30.75dB**

Table 4. Study on the impact of adaptive scaling on model performance (x4).

Case	PSNR(dB)	Parame.(K)	GPU Memory
w/o TR	31.96	554	1931M
Original TR [22]	32.14	971	16057M
1 ET	**32.18**	751	4191M
2 ET	32.25	949	6499M
s=1	32.14	751	13580M
s=2	32.15	751	6731M
s=4	**32.18**	751	4191M
s=6	32.04	751	3159M

Table 5. Study of Efficient Transformer (ET) on Set5 ($\times 4$). The GPU memory here refers to the cost of the model during training, which patch_size = 48*48 and batch_size=16.

1.3. Study of Adaptive Scaling

ARFB contains two Residual Units (RUs) and two convolutional layers, which is one of the most basic components to build the High Preserving Block (HPB). Meanwhile, a residual scaling with adaptive weights (RSA) is designed to dynamically adjust the importance of residual path and identity path. To verify the effectiveness of the adaptive scaling mechanism, we provide a ablation study in Table 4. Among them, X and $\surd$ represent the models with and without the adaptive scaling mechanism in ARFB, respectively. Obviously, with the help of the adaptive scaling mechanism, the performance of the model can be further improved. This fully demonstrates the effectiveness of the adaptive scaling mechanism.

1.3.1 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). To illustrate the efficiency and effectiveness of ET, we provide the following experiments:

A. **TR v.s. w/o TR**: Firstly, we analyze the model with and without Transformer in TABLE 5. We can see that if we remove the Transformer, the model performance descends obviously from 32.18dB to 31.96dB. From this case, it can be inferred that the correlation of long-term image patches is beneficial for image super-resolution. The reason is that a natural scene image has many similar pixel blocks and these blocks always can complete other missing information as a reference. Therefore, the introduced Transformer can make full advantage of this relationship.

B. **ET v.s. Original TR**: Secondly, we compare our ET with the original Transformer in computer vision (ViT [8]). From TABLE 5 we can see that for the original TR, it will increase 417M parameters while our ET (1 ET) only increases 197M parameters. This benefits from the Reduction module that can reduce the number of channels. In addition, for GPU memory, the original TR occupies 16057M memory which even cannot run on some common NVIDIA GPUs like 1080Ti and 2080Ti. Contrastly, our ET just occupies 4191M GPU memory, which is only 1/4 of the original Transformer. More surprising is that the performance of the model with the original Transformer is even worse than our ESRT (1 ET). This is because the model with the original Transformer needs more data to train while the datasets are usually small in the SISR task. This experiment further verified the effectiveness of our proposed ET.

C. **The Number of ET**: In general, increasing the number of convolutional layers can increase the model performance. In view of this, we also added the number of ET in our model to explore its performance. From TABLE 5, we can 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 keep consistent with other lightweight models in the aspect of parameters, only one ET is used in the final ESRT.

D. **The Splitting Factor** s: In MHA, a Feature Split Module (FSM) is used to split the original Q, K, and V into s segments to save the GPU memory. Commonly, the s is larger, the split segments are shorter and the GPU memory occupation is less. In TABLE 5, we investigate the different value of s. Obviously, the model achieves the best performance when $s = 4$. Meanwhile, we can observe that the change of s will not affect the number of model parameters. Therefore, we set $s = 4$ in the final model.

Model	Parameter	GPU occupy	Set5	Set14	BSD100	Urban100	Manga109
			PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR / SSIM
ESRT (Ours)	751K	4191M	32.19/0.8947	28.69/0.7833	27.69/0.7379	26.39/0.7962	30.75/0.9100
Pure ESRT (1ET)	357K	3967M	31.01/0.8751	27.85/0.7636	27.10/0.7203	25.00/0.7459	28.22/0.8726
Pure ESRT (2ET)	564K	5685M	31.77/0.8878	28.39/0.7758	27.42/0.7312	25.73/0.7728	29.76/0.8978
Pure ESRT (3ET)	771K	7409M	32.10/0.8926	28.59/0.7808	27.57/0.7360	26.13/0.7853	30.32/0.9057
Pure ESRT (4ET)	978K	9121M	32.29/0.8948	28.71/0.7830	27.64/0.7384	26.42/0.7936	30.69/0.9109
Pure ESRT (6ET)	1392K	12647M	32.36/0.8965	28.80/0.7850	27.70/0.7405	26.69/0.8016	30.97/0.9135
Pure ESRT (8ET)	1806K	16163M	32.40/0.8751	28.84/0.7858	27.73/0.7412	26.83/0.8048	31.11/0.9146
SAN [5]	15700K	12912M	32.64/0.9003	28.92/0.7888	27.78/0.7436	26.79/0.8068	31.18/0.9169

Table 6. Quantitative comparison between our ESRT, Pure ESRT, and SAN [5] ($\times 4$). The GPU memory here refers to the cost of the model during training, which patch_size = 48*48 and batch_size=16.

Scale	Bicubic		SRCNN [6]		VDSR [11]		SRResNet [14]		IMDN [9]		LK-KPN [3]		ESRT (Ours)	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
$\times 2$	32.61	0.907	33.40	0.916	33.64	0.917	33.69	0.919	33.85	0.923	-	-	**33.92**	**0.924**
$\times 3$	29.34	0.841	29.96	0.845	30.14	0.856	30.18	0.859	30.29	0.857	**30.60**	**0.863**	30.38	0.857
$\times 4$	27.99	0.806	28.44	0.801	28.63	0.821	28.67	0.824	28.68	0.815	28.65	**0.820**	**28.78**	0.815

Table 7. PSNR and SSIM comparison with other advanced SISR methods on the RealSR dataset.

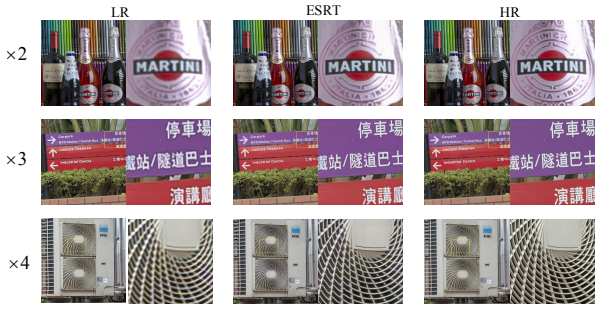

Figure 2. Visual comparison on RealSR dataset (version3). The SR images reconstructed by our ESRT have more accurate edges.

1.4. Study of the Pure Transformer

In general, the pure Transformer-based architecture is more efficient and scalable than previous CNN-based architecture in both model size and computational cost. However, we find that the hybrid Transformer can perform better than the pure Transformer model on a lightweight model. To verify this view, we provide the performance of pure Transformer-based ESRT. Specifically, we modify ESRT to the pure Transformer by removing the LCB. We define the modified Transformer as "Pure ESRT". According to TABLE 6, it can be seen that if the number of ET in LTB is set to 1 in both Pure ESRT and ESRT, the performance of Pure ESRT will significantly decrease compared with ESRT. This means that LCB can effectively compensate for the lack of feature extraction capabilities of the Transformer. When the number of ETs in pure Transformer is increased to 3, the parameters of the model are close to our ESRT, but its performance is not as good as our ESRT and it will take up more GPU memory. This fully demonstrates the effectiveness of our proposed hybrid Transformer.

Meanwhile, we can see that our "Pure ESRT (8ET)" achieves close performance compared with the state-of-the-art method SAN [5] with only one-ninth parameters. Moreover, our model even achieves better performance on Urban100 than SAN. This reflects that building Pure-ESRT can achieve comparable SR performance compared with a well-designed CNN model.

1.5. Real Image Super-Resolution

In this part, we compare our ESRT with more classic lightweight SR models (e.g., SRCNN [6], VDSR [11], SRResNet [14], IMDN [9] and LK-KPN [3]) on the real image dataset (RealSR [3]). It is worth noting that since the resolution of the LR and HR images is the same in RealSR, the PixelShuffle is removed in our model and only one convolutional layer is applied to change the feature map into SR images. According to TABLE 7, we can observe that compared to IMDN, the performance of ESRT gains 0.07dB, 0.09dB, and 0.10dB for scaling factors $\times 2$, $\times 3$, and $\times 4$, respectively. Also, our model has a close performance to LK-KPN which was specifically designed for the RealSR task. In addition, we provide the reconstructed SR images in Figure 2. Obviously, our ESRT recovers line edges effectively, such as some Chinese words and English words. Meanwhile, our ESRT can restore the texture details well, such as the grid lines in the air conditioner. All these ex-

Method	Dataset	Parame.	GPU Memory	Set5	Set4	BSD100	Urban00	Manga109
SwinIR	DIV2k + Flickr2K	897K	6966M	32.44/0.8976	28.77/0.7858	27.69/0.7406	26.47/0.7980	30.92/0.9151
ESRT	DIV2k	751K	4191M	32.19/0.8947	28.69/0.7833	27.69/0.7379	26.39/0.7962	30.75/0.9100

Table 8. A detailed comparison of SwinIR and our ESRT ($\times 4$). The GPU memory here refers to the cost of the model during training, which patch_size = 48*48 and batch_size=16.

periments show that our ESRT can also obtain a good SR property in the real world.

1.6. Different from other Transformer-base Methods

In recent years, some Transformer-base methods have been proposed for low-level image processing tasks. For example, In IPT [4], a novel Transformer-based network is proposed as the pre-trained model for low-level image restoration tasks. However, IPT utilizes ImageNet (more than 1.3M images) for training and has a huge number of parameters (115.5M), which is difficult to use in practical applications. In [23], RefSR uses different patches as the Q, K, and V to fuse various information of the reference images. Our method also follows this idea but the difference is that similar patches of ESRT are explored in the original image itself. Meanwhile, The complexity and computational cost of ESRT are lower than RefSR. In [15], a SwinIR is proposed for image restoration. The EMHA in our ESRT is similar to the Swin Transformer layer of SwinIR. 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.

In addition, we selected the most representative SwinIR for comparison. We use the official code and test it on our server with the same environment and setting. The results are provided in Table 8. Obviously, our ESRT achieves close performance to SwinIR 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. All these results further validates the effectiveness of the proposed ESRT.

2. Discussions

Benefits of LCB. LCB solves the problem of the poor feature extraction ability of Transformer on small datasets. It is a lightweight architecture that can efficiently extract deep SR features. Meanwhile, LCB can be easily embedded into any SISR model to reduce model parameters and calculation costs, and maintain good performance.

Benefits of LTB. LTB solves the problem of heavy GPU memory consumption in vision Transformer. Meanwhile, ET can model the dependence between long-term sub-image blocks in the LR, enhancing the structural information of every image region. It has been improved that model such a long-term dependency of similar local regions is helpful for SR task. Meanwhile, ET is a lightweight and universal module that can be embedded into any present SR model to further improve model performance.

Limitations of ESRT. In this work, we propose a hybrid architecture consisting of CNN and Transformer. In order to keep the low complexity of the model, we directly connect the Transformer after the CNN. Although our experiments have verified the effectiveness of this method, we believe that there are more effective methods that can better utilize the local features extracted by CNN and the global relationship learned by Transformer. In future works, we will explore more effective combining methods to further improve the performance of the model.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017. 6

[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. 2

[3] 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. 5

[4] 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. 6

[5] 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. 5

[6] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE TPAMI*, 38(2):295–307, 2015. 2, 5

[7] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, pages 391–407, 2016. 2

[8] 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. 4

[9] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACMMM*, pages 2024–2032, 2019. 2, 5

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

[11] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016. 2, 5

[12] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, pages 1637–1645, 2016. 2

[13] 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. 2

[14] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, pages 4681–4690, 2017. 5

[15] 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. 6

[16] 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. 2

[17] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *ECCV*, pages 41–55, 2020. 2

[18] 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

[19] 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. 2

[20] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *CVPR*, pages 3147–3155, 2017. 2

[21] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, pages 4539–4547, 2017. 2

[22] 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. 4

[23] 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. 6

[24] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *CVPR*, pages 3262–3271, 2018. 2