Emulating Self-attention with Convolution for Efficient Image Super-Resolution

Supplementary Material

The supplementary includes implementation details of Flash Attention (FA) and proposed networks, additional results on the arbitrary-scale super-resolution (SR) tasks, quantitative results on real-world SR tasks, efficiency comparison beyond GPUs, and classic SR results on larger model sizes.

6. Flash Attention Implementation Details

FA [10] implements self-attention by fusing kernels and avoiding materializing a full score matrix ($S = QK^T$), thereby significantly reducing both memory footprint and latency. Although FA has been widely adopted across various domains, such as classification and generation, its application to SR tasks has been limited. In order to alleviate the self-attention's memory bottleneck, we attempted to integrate FA directly into SR architectures. However, we observed that naively applying FA leads to highly unstable training. We verified that this instability stems from FA's fused-kernel design, which blocks the use of relative positional bias (RelPos) [40, 49]. As shown in Table 6, training loss diverges when FA is employed without RelPos. A common alternative is to use Rotary Positional Encoding (RoPE) [23, 51], which rotates query and key tensors for positional conditioning and therefore remains compatible with fused kernels [65, 66]. Nonetheless, our experiments demonstrate that even this approach fails to deliver acceptable performance in the SR setting. To address these challenges, we leverage Flex Attention [14], which supports both user-defined score modifications and FA. Our implementation not only resolves the memory bottleneck but also achieves superior performance by leveraging the 32×32 window size for self-attention.

Table 6. Comparisons on performance by FA implementation.

Type	FA [10]	FA w/ RoPE [23]	**FA w/ RelPos (Ours)**
PSNR/SSRM (U100×2)	NaN / NaN	32.76 / 0.9343	**33.46 / 0.9395**

7. Network Implementation Details

This section describes the details of the implementation of our methods. We begin by outlining the basic model configuration for each task, which includes the hyperparameters C, N, and M. Here, C represents the number of channels, N denotes the number of $\mathrm{ESCBlocks}$, and M indicates the number of $\mathrm{ConvAttns}$. Next, we describe additional components, such as the compositions of U and S, among others. Finally, we present the training details, which cover the training datasets, the number of training iterations, the learning rate, optimizer configurations, the loss function used, and the training batch and patch size.

7.1. Classic SR

Three variants are introduced in the classic SR task: ESC-FP, ESC-light, and ESC. ESC-FP is a variant needed when it is necessary to reduce FLOPs and parameter size, while ESC-light and ESC are variants needed when it is necessary to reduce latency. The basic configurations for each variant are as follows: for ESC-FP, C, N, and M are 48, 5, and 5; for ESC-light, they are 64, 3, and 5; and for ESC, they are 64, 5, and 5. All variants use Sub-Pixel Convolution (SPConv) [50] as U. ESC-light and ESC utilize the repeat function [16] as S and add F^s before the pixel shuffle of SPConv, while ESC-FP employs bicubic interpolation as S and adds F^s to the output of SPConv. Furthermore, ESC-FP employs decomposed LK to further reduce FLOPs and parameter size and utilizes extra layer normalizations in front of the $\mathrm{ConvFFNs}$. Lastly, hidden dimension (h) for kernel estimators is set to 8 for ESC and ESC-FP, while ESC-FP uses 4. For training, we use the DIV2K [1] dataset, and for the data scaling experiments, we employ the DIV2K+Flickr2K+LSDIR+Diverseg-IP (DFLIP) [34, 45, 54] dataset. Training our networks lasts for 500K iterations, and the optimizer used is AdamW [41] with βs of 0.9 and 0.9 and a learning rate of 5e-4. We use L1 loss and 64 patches of size 64×64 as input to train. The networks of scale ×3 and ×4 are fine-tuned from the results of the ×2. To train other methods [73, 76, 82] for data scaling experiments, we follow the training details described in their paper.

7.2. Arbitrary-scale SR

For the Arbitrary-scale SR task, we use the same basic configuration as the ESC. The difference between the ESC in Classic SR and the ESC in arbitrary-scale SR is that LTE [32] is used as U, and accordingly, S is also changed to bilinear interpolation. Training details, including other models [73, 76], are the same as LTE across all instances, using the DIV2K dataset, running for 1000 epochs, utilizing the Adam [30] with βs of 0.9 and 0.999, and leveraging L1 loss. However, since HiT-SRF and our ESC are optimized for training with the input patches of 64×64, we train all Transformers leveraging 32 input patches of size 64×64. Still, the number of sampling coordinates for training remains the same as RDN+LTE, which is 2304 (48^2).

Table 7. Quantitative comparison on arbitrary-scale SR task employing LTE [32] as upsampler. The best result on PSNR is bolded.

Methods	Set5					Set14					B100					Urban100				
	Seen			Unseen		Seen			Unseen		Seen			Unseen		Seen			Unseen	
	×2	×3	×4	×6	×8	×2	×3	×4	×6	×8	×2	×3	×4	×6	×8	×2	×3	×4	×6	×8
RDN+LTE [78]	38.23	34.72	32.61	**29.32**	**27.26**	**34.09**	30.58	28.88	**26.71**	25.16	32.36	29.30	**27.77**	**26.01**	24.95	33.04	28.97	26.81	24.28	22.88
ATD-lt+LTE§ [73]	38.28	34.73	32.57	29.21	27.22	34.14	30.64	28.91	26.67	25.21	32.35	29.30	**27.77**	**26.01**	24.95	33.12	29.06	26.95	24.41	23.00
HiT-SRF+LTE§ [76]	38.27	34.74	32.59	29.25	27.21	34.03	30.64	28.91	26.68	25.21	32.37	29.30	27.76	26.00	24.94	33.18	29.05	26.89	24.34	22.94
ESC+LTE (Ours)	**38.29**	**34.79**	**32.72**	29.18	27.24	34.05	**30.69**	**28.94**	26.70	**25.24**	**32.38**	**29.32**	**27.77**	26.00	**24.96**	**33.30**	**29.21**	**27.04**	**24.44**	**23.03**

7.3. Real-world SR

For the real-world SR task, we introduce ESC-Real with a basic configuration of 64, 10, and 5, which denote C, N, and M, respectively. ESC-Real employs the same upsampler as RealESRGAN [59] and SwinIR-Real [36], utilizing it as U, and incorporates four layers of shallow network (`c128k1g1–c128k7g128–LeakyReLU(`$\alpha = 0.2$`)–c64k1g1`) as S. Here, `c`, `k`, and `g` denote channel, kernel, and group size for convolution, respectively. In this approach, F^s is added into F. We use the RealESRGAN degradation model and DF2KOST [58] dataset to generate low-quality images. ESC-real is first trained for 1M iterations using L1 loss, then trained for 400K iterations with L1 loss, adversarial loss, and perceptual loss, using weight factors of 1, 0.1, and 1, respectively. The network architectures used for calculating adversarial loss and perceptual loss are the same as those used in RealESRGAN. In both phases, 48 patches of size 64×64 are used as input for training.

8. Additional Results on Arbitrary-scale SR

This section exhibits additional results on the arbitrary-scale SR task. Additional quantitative results are measures on the four commonly used evaluation datasets, including Set5 [3], Set14 [70], B100 [42], and Urban100 [26]. Following previous research [32], we measure Peak Signal-to-Noise Ratio (PSNR) on the Y channel after cropping the image's boundary equivalent to the upscaling factor and converting it to YCbCr color space. Our ESC+LTE outperforms other methods on Urban100 at both seen and unseen scales, as shown in Table 7.

9. Quantitative Results on Real-world SR

Taking a step further, we report quantitative results for the real-world SR task. To this end, we measure a variety of metrics (PSNR@Y, SSIM@Y, LPIPS [74], DISTS [11], FID [24], NIQE [44], MANIQA [64], MUSIQ [27], and CLIP-IQA [57]) on multiple datasets (RealLQ250 [2], RealSRSet [72], RealSR [4], and DRealSR [60]). As shown in Table 8, ESC-Real achieves the highest CLIPIQA scores on all datasets, demonstrating its ability to reconstruct perceptually superior images.

Table 8. Quantitative comparisons on real-world SR.

Dataset	Metrics	RealESRGAN+	SwinIR-Real	DASR	**ESC-Real**
RealLQ250 [2]	NIQE ↓	4.1328	4.1779	4.7858	4.0556
	MANIQA ↑	0.3564	0.3400	0.2789	0.3553
	MUSIQ ↑	62.51	60.48	53.02	62.98
	CLIPIQA ↑	0.5437	0.5348	0.4631	0.5796
RealSRSet [72]	NIQE ↓	5.3430	5.1037	4.5931	5.0181
	MANIQA ↑	0.3988	0.3872	0.3277	0.3952
	MUSIQ ↑	64.25	63.68	58.82	64.58
	CLIPIQA ↑	0.5942	0.5921	0.5278	0.6156
RealSR [4]	PSNR ↑	24.53	24.71	25.86	24.52
	SSIM ↑	0.7484	0.7547	0.7617	0.7503
	LPIPS ↓	0.2729	0.2594	0.3113	0.2622
	DISTS ↓	0.1685	0.1609	0.1838	0.1671
	FID ↓	67.01	64.19	63.62	66.81
	NIQE ↓	4.6801	4.6465	5.9682	4.4848
	MANIQA ↑	0.3675	0.3504	0.2663	0.3799
	MUSIQ ↑	59.69	59.64	45.82	61.40
	CLIPIQA ↑	0.4903	0.4736	0.3629	0.5338
DRealSR [60]	PSNR ↑	26.59	26.52	28.40	26.76
	SSIM ↑	0.7988	0.7923	0.8302	0.79534
	LPIPS ↓	0.2818	0.2838	0.2962	0.2795
	DISTS ↓	0.1464	0.1461	0.1689	0.1503
	FID ↓	23.19	24.63	17.89	21.35
	NIQE ↓	4.7164	4.5683	6.3473	4.7006
	MANIQA ↑	0.3431	0.3275	0.2733	0.3442
	MUSIQ ↑	35.27	34.62	28.63	35.21
	CLIPIQA ↑	0.5179	0.5039	0.3843	0.5564

Table 9. Comparisons of latency on MacBook M2 air, and iPhone 12

Methods	ELAN-lt	OmniSR	ASID-D8	HiT-SRF	**ESC-lt**	**ESC**
M2Air ($X \in \mathbb{R}^{128\times128\times3}$)	318.51	Failed	Failed	88145.85	**124.07**	**181.16**
iPhone12 ($X \in \mathbb{R}^{32\times32\times3}$)	38.12	Failed	Failed	OOM	**25.57**	**42.78**

10. Efficiency Comparisons beyond GPUs

In real-world deployment scenarios, networks often run on devices with far more constrained resources than GPUs. To evaluate our method under such conditions, we benchmarked several Transformer-based SR models (ELAN [75], OmniSR [56], ASID-D8 [47], HiT-SRF [76]) against our ESC(-lt) on a MacBook Air M2 and an iPhone 12. As detailed in Table 9, whereas the other Transformers either fail to compile or incur out-of-memory (OOM) errors, ESC-lt achieves up to a 61% reduction in latency compared to ELAN-light, demonstrating its efficiency in real-world deployments.

Table 10. Comparisons of larger classic SR methods (Params>10M). PT denotes pre-training with 64×64 patches and FT denotes fine-tuning with 96×96 patches.

Method	Scale	#Params (M)	PSNR / SSIM Set5	Set14	B100	Urban100	Manga109
SwinIR [36]	×2	11.8	38.42/0.9623	34.46/0.9250	32.53/0.9041	33.81/0.9433	39.92/0.9797
EDT-B [33]		11.5	38.63/0.9632	34.80/0.9273	32.62/0.9052	34.27/0.9456	40.37/0.9811
CAT-A [9]		16.5	38.51/0.9626	34.78/0.9265	32.59/0.9047	34.26/0.9440	40.10/0.9805
ART [71]		16.4	38.56/0.9629	34.59/0.9267	32.58/0.9048	34.30/0.9452	40.24/0.9808
ACT [67]		46.0	38.46/0.9626	34.60/0.9256	32.56/0.9048	34.07/0.9443	39.95/0.9804
SRFormer [82]		10.5	38.51/0.9627	34.44/0.9253	32.57/0.9046	34.09/0.9449	40.07/0.9802
ESC (PT)		12.5	38.52/0.9626	34.57/0.9257	32.58/0.9045	34.24/0.9450	40.18/0.9803
ESC (FT)		12.5	38.59/0.9630	34.70/0.9259	32.61/0.9052	34.49/0.9466	40.38/0.9809
SwinIR [36]	×3	11.9	34.97/0.9318	30.93/0.8534	29.46/0.8145	29.75/0.8826	35.12/0.9537
EDT-B [33]		11.7	35.13/0.9328	31.09/0.8553	29.53/0.8165	30.07/0.8863	35.47/0.9550
CAT-A [9]		16.6	35.06/0.9326	31.04/0.8538	29.52/0.8160	30.12/0.8862	35.38/0.9546
ART [71]		16.6	35.07/0.9325	31.02/0.8541	29.51/0.8159	30.10/0.8871	35.39/0.9548
ACT [67]		46.0	35.03/0.9321	31.08/0.8541	29.51/0.8164	30.08/0.8858	35.27/0.9540
SRFormer [82]		10.7	35.02/0.9323	30.94/0.8540	29.48/0.8156	30.04/0.8865	35.26/0.9543
ESC (FT)		12.5	35.14/0.9330	31.10/0.8552	29.53/0.8167	30.23/0.8895	35.60/0.9555
SwinIR [36]	×4	11.9	32.92/0.9044	29.09/0.7950	27.92/0.7489	27.45/0.8254	32.03/0.9260
EDT-B [33]		11.6	33.06/0.9055	29.23/0.7971	27.99/0.7510	27.75/0.8317	32.39/0.9283
CAT-A [9]		16.6	33.08/0.9052	29.18/0.7960	27.99/0.7510	27.89/0.8339	32.39/0.9285
ART [71]		16.6	33.04/0.9051	29.16/0.7958	27.97/0.7510	27.77/0.8321	32.31/0.9283
ACT [67]		46.0	32.97/0.9031	29.18/0.7954	27.95/0.7507	27.74/0.8305	32.20/0.9267
SRFormer [82]		10.6	32.93/0.9041	29.08/0.7953	27.94/0.7502	27.68/0.8311	32.21/0.9271
ESC (FT)		12.5	33.00/0.9054	29.21/0.7968	27.95/0.7504	27.89/0.8351	32.54/0.9295

11. Classic SR Results on Larger Model Size

Although we have demonstrated substantial performance gains over lightweight Transformers (Params<1M), larger models (Params>10M) remain an active area of research. To assess our method in this regime, we scale ESC to 12.5M parameters, on par with the size of SwinIR, and train and evaluate it accordingly. The scaled ESC uses the window size of 48×48, $N = 8$, $M = 5$, $C = 192$, $C_{\mathrm{ConvAttn}} = 48$, and $h = 24$. Extra layer normalizations are placed before the $\mathrm{ConvFFN}$s. We leverage the DF2K dataset and follow the ATD's training strategy [73], pre-training on small patches (64×64) and then fine-tuning on larger patches (96×96). Table 10 shows that ESC delivers performance on par with other large-scale SR Transformers.