Supplementary material for "LIRA: Lifelong Image Restoration from Unknown Blended Distortion"

Jianzhao Liu*[0000−0002−5811−0881], Jianxin Lin*, Xin Li[0000−0002−6352−6523], Wei Zhou[0000−0003−3641−1429], Sen Liu[0000−0003−3778−8973], and Zhibo Chen†[0000−0002−8525−5066]

CAS Key Laboratory of Technology in Geo-spatial Information Processing and Application System,
University of Science and Technology of China, Hefei 230027, China
{jianzhao,linjx,lixin666,weichou}@mail.ustc.edu.cn
elsen@iat.ustc.edu.cn, chenzhibo@ustc.edu.cn

1 Overview

In this supplementary material, we provide additional results to complement the paper: (1) We provide the network parameters and configuration of our base network, our expanded network, the generator and the discriminator in Section 2. (2) We show comparisons with the state-of-the-art algorithms in terms of complexity in Section 3.1. (3) We show qualitative results of pseudo-samples generated by the generator in Section 3.2.

2 Network Architectures

Our incremental network consists of the base network and the expanded network. We also employ a generator that generates pseudo samples to consolidate the knowledge of the old task. Table 1 and Table 2 list the detailed configuration and parameters of our base network and our expanded network respectively. Table 3 shows the architecture of the generator and Table 4 shows the architecture of the discriminator.

Table 1: **Detailed architecture of our base network.** It consists of a feature extractor, a feature amalgamator and a decoder. $[\cdot]$ refers to concatenation operation. The feature amalgamator consists of 3 experts, a set of gate layers, an attention layer, an aggregation layer and two Bidirectional Sequential Gateing Units (Bi-SGUs). In the decoder, we use PixelShuffle [1] module as the upsampling layer to upsample the feature maps by $2\times$. The final reconstructed image is the summation of the output of the *Decoder*1 (out_up1-2) and the input image.

*The first two authors contributed equally to this work.
†Corresponding author.

Base network		Input	Output	Kernel size	Input channels	Output channels	Stride	Output size
Feature extractor1	conv1-1	image1 (RGB)	global_feature1-1	3×3	3	32	1	64×64
	conv1-2	global_feature1-1	global_feature1-2	3×3	32	32	2	32×32
	conv1-3	global_feature1-2	global_feature1-3	3×3	32	32	2	16×16
Expert1	expert1_ SK-DuRB1	global_feature1-2, global_feature1-3	expert1_res1, expert1_out1	-	32, 32	32, 32	-	32×32, 16×16
	expert1_ SK-DuRB2	expert1_res1, expert1_out1	expert1_res2, expert1_out2	-	32, 32	32, 32	-	32×32, 16×16
	...	...	...	...	...	...	...	...
	expert1_ SK-DuRB6	expert1_res5, expert1_out5	expert1_res6, expert1_out6	-	32, 32	32, 32	-	32×32, 16×16
Expert2	expert2_ SK-DuRB1	global_feature1-2, global_feature1-3	expert2_res1, expert2_out1	-	32, 32	32, 32	-	32×32, 16×16
	expert2_ SK-DuRB2	expert2_res1, expert2_out1	expert2_res2, expert2_out2	-	32, 32	32, 32	-	32×32, 16×16
	...	...	...	...	...	...	...	...
	expert2_ SK-DuRB6	expert2_res5, expert2_out5	expert2_res6, expert2_out6	-	32, 32	32, 32	-	32×32, 16×16
Expert3	expert3_ SK-DuRB1	global_feature1-2, global_feature1-3	expert3_res1, expert3_out1	-	32, 32	32, 32	-	32×32, 16×16
	expert3_ SK-DuRB2	expert3_res1, expert3_out1	expert3_res2, expert3_out2	-	32, 32	32, 32	-	32×32, 16×16
	...	...	...	...	...	...	...	...
	expert3_ SK-DuRB6	expert3_res5, expert3_out5	expert3_res6, expert3_out6	-	32, 32	32, 32	-	32×32, 16×16
Attention layer1		global_feature1-3	weightsA/weightsB	-	32	1	-	3×3
3×Gate layer (Expert1)		[image1, global_feature1-2, expert1_res2/ expert1_res4/ expert1_res6]	expert1_gate_res2/ expert1_gate_res4/ expert1_gate_res6	3×3	67	32	1	32×32
3×Gate layer (Expert1)		[image1, global_feature1-3, expert1_out2/ expert1_out4/ expert1_out6]	expert1_gate_out2/ expert1_gate_out4/ expert1_gate_out6	3×3	67	32	1	16×16
3×Gate layer (Expert2)		[image1, global_feature1-2, expert2_res2/ expert2_res4/ expert2_res6]	expert2_gate_res2/ expert2_gate_res4/ expert2_gate_res6	3×3	67	32	1	32×32

3×Gate layer (Expert2)		[image1, global_feature1-3, expert2_out2/ expert2_out4/ expert2_out6]	expert2_gate_out2/ expert2_gate_out4/ expert2_gate_out6	3×3	67	32	1	16×16
3×Gate layer (Expert3)		[image1, global_feature1-2, expert3_res2/ expert3_res4/ expert3_res6]	expert3_gate_res2/ expert3_gate_res4/ expert3_gate_res6	3×3	67	32	1	32×32
3×Gate layer (Expert3)		[image1, global_feature1-3, expert3_out2/ expert3_out4/ expert3out6]	expert3_gate_out2/ expert3_gate_out4/ expert3_gate_out6	3×3	67	32	1	16×16
Aggregation layer1		weightsA, weightsB, expert$n$_gate_res$i$, expert$n$_gate_out$i$, expert$n$_res$i$, expert$n$_out$i$ (n=1,2,3;i=2,4,6)	aggregated1_res2, aggregated1_res4, aggregated1_res6, aggregated1_out2, aggregated1_out4, aggregated1_out6	-	-	32, 32, 32, 32, 32, 32	-	32×32, 32×32, 32×32, 16×16, 16×16, 16×16
Bi-SGU1-1		aggregated1_res2, aggregated1_res4, aggregated1_res6,	res1	3×3	32, 32, 32	32	1	32×32
Bi-SGU1-2		aggregated1_out2, aggregated1_out4, aggregated1_out6	out1	3×3	32, 32, 32	32	1	16×16
conv1-4		res1	res1	3×3	32	32	2	16×16
conv1-5		[res1,out1]	out_final1	3×3	64	32	1	16×16
Decoder1	Upsampling layer1-1	out_final1	out_upl-1	3×3	32	32	1/2	32×32
	Upsampling layer1-2	out_upl-1	out_up1-2	3×3	32	32	1/2	64×64

Table 2: **Detailed architecture of our expanded network.** We keep the intact architecture of the base network and all parts except Bi-SGUs in the amalgamator of the base network are treated as a whole and named as an old expert. The final reconstructed image is the summation of the output of the *Decoder*2 (out_up2-2) and the input image.

Expanded network		Input	Output	Kernel size	Input channels	Output channels	Stride	Output size
Feature extrator2	conv2-1	image2 (RGB)	global_feature2-1	3×3	3	32	1	64×64
	conv2-2	global_feature2-1	global_feature2-2	3×3	32	32	2	32×32
	conv2-3	global_feature2-2	global_feature2-3	3×3	32	32	2	16×16

Old expert		global_feature2-2, global_feature2-3	old_aggregated_res2, old_aggregated_res4, old_aggregated_res6, old_aggregated_out2, old_aggregated_out4, old_aggregated_out6	-	32, 32	32, 32, 32, 32, 32, 32	-	32×32, 32×32, 32×32, 16×16, 16×16, 16×16
Expert4	expert4_ SK-DuRB1	global_feature2-2, global_feature2-3	expert4_res1, expert4_out1	-	32, 32	32, 32	-	32×32, 16×16
	expert4_ SK-DuRB2	expert4_res1, expert4_out1	expert4_res2, expert4_out2	-	32, 32	32, 32	-	32×32, 16×16
	...	...	...	...	...	...	...	...
	expert4_ SK-DuRB6	expert4_res5, expert4_out5	expert4_res6, expert4_out6	-	32, 32	32, 32	-	32×32, 16×16
Attention layer2		global_feature2-3	weightsC/weightsD	-	32	1	-	3×3
3×Gate layer (old expert)		[image2, global_feature2-2, old_aggregated_res2/ old_aggregated_res4/ old_aggregated_res6]	oldexpert_gate_res2/ oldexpert_gate_res4/ oldexpert_gate_res6	3×3	67	32	1	32×32
3×Gate layer (old expert)		[image2, global_feature2-3, old_aggregated_out2/ old_aggregated_out4/ old_aggregated_out6]	oldexpert_gate_out2/ oldexpert_gate_out4/ oldexpert_gate_out6	3×3	67	32	1	16×16
3×Gate layer (Expert4)		[image2, global_feature2-2, expert4_res2/ expert4_res4/ expert4_res6]	expert4_gate_res2/ expert4_gate_res4/ expert4_gate_res6	3×3	67	32	1	32×32
3×Gate layer (Expert4)		[image2, global_feature2-3, expert4_out2/ expert4_out4/ expert4_out6]	expert4_gate_out2/ expert4_gate_out4/ expert4_gate_out6	3×3	67	32	1	16×16
Aggregation layer2		weightsC, weightsD, oldexpert_gate_res$i$, oldexpert_gate_out$i$, expert4_gate_res$i$, expert4_gate_out$i$, oldexpert_res$i$, oldexpert_out$i$, expert4_res$i$, expert4_out$i$ (i=2,4,6)	aggregated2_res2, aggregated2_res4, aggregated2_res6, aggregated2_out2, aggregated2_out4, aggregated2_out6	-	-	32, 32, 32, 32, 32, 32	-	32×32, 32×32, 32×32, 16×16, 16×16, 16×16
Bi-SGU2-1		aggregated2_res2, aggregated2_res4, aggregated2_res6	res2	3×3	32, 32, 32	32	1	32×32

Bi-SGU2-2		aggregated2_out2, aggregated2_out4, aggregated2_out6	out2	3×3	32, 32, 32	32	1	16×16
conv2-4		res2	res2	3×3	32	32	2	16×16
conv2-5		[res2,out2]	out_final2	3×3	64	32	1	16×16
Decoder2	Upsampling layer2-1	out_final2	out_up2-1	3×3	32	32	1/2	32×32
	Upsampling layer2-2	out_up2-1	out_up2-2	3×3	32	32	1/2	64×64

Table 3. Detailed architecture of the generator. We use the transposed convolution to upsample the feature maps by $2\times$.

Layer	Kernel size	Input channels	Output channels	Stride	Output size
conv1	7×7	3	64	1	64×64
conv2	3×3	64	128	2	32×32
conv3	3×3	128	256	2	16×16
9×ResBlock	3×3	256	256	1	16×16
upsampling1	3×3	256	128	1/2	32×32
upsampling2	3×3	128	64	1/2	64×64
conv4	7×7	64	3	1	64×64

Table 4. Detailed architecture of the discriminator. The input is a 64×64 RGB image. The output is a single scalar.

Layer	Kernel size	Input channels	Output channels	Stride	Output size
conv1	4×4	3	64	2	32×32
conv2	4×4	64	128	2	16×16
conv3	4×4	128	256	2	8×8
conv4	4×4	256	512	1	8×8
conv5	4×4	512	1	1	8×8
Average pooling	8×8	1	1	8×8	1×1

3 Additional Experimental Results

3.1 Comparisons in terms of complexity

The complexity of RL-Restore [3], OWAN [2], our base network and our expanded network can be found in Table 5. We compute the complexity of these methods on a 63×63 image. It can be seen that the complexity of our base network is lower than that of OWAN. In addition, the complexity of the expanded network increases by just 33.4% over that of the base network. In contrast, if we train two individual networks for two tasks respectively, the overall complexity of the networks is twice as that of the base network.

Table 5. Comparisons with DnCNN, RL-Restore and OWAN in terms of complexity.

Methods	RL-Restore [3]	OWAN [2]	Our base network	Our expanded network
FLOPs(G)	0.474	1.439	1.259	1.679

3.2 Qualitative results of pseudo samples

For training our incremental network, pseudo samples generated by the generator are paired with the corresponding responses of our trained base network, serving as supervision of the old task. We show examples of real and pseudo samples with corresponding restored samples generated by our trained base model in Fig. 1. We can see that the generator can resemble the data distribution of the real samples to some extent, playing the role of memory replay to consolidate the knowledge of the old task.

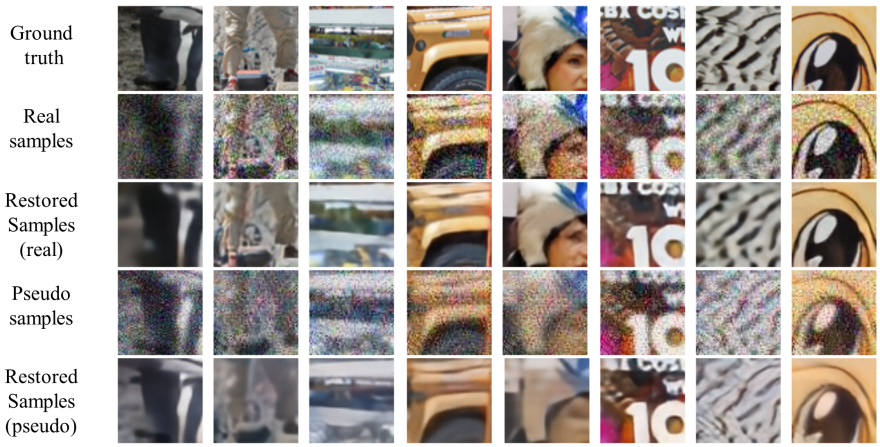

Fig. 1. Qualitative results of real and pseudo samples with corresponding restored samples generated by our trained base model.

References

1. Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient subpixel convolutional neural network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
2. Suganuma, M., Liu, X., Okatani, T.: Attention-based adaptive selection of operations for image restoration in the presence of unknown combined distortions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9039–9048 (2019)
3. Yu, K., Dong, C., Lin, L., Change Loy, C.: Crafting a toolchain for image restoration by deep reinforcement learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2443–2452 (2018)