Modular Degradation Simulation and Restoration for Under-Display Camera
Supplementary Materials

Yang Zhou, Yuda Song, and Xin Du ✉

Zhejiang University, Hangzhou, China
{yang_zhou,syd,duxin}@zju.edu.cn

A Discriminator

Considering that our generation process is an image-to-image task, we define a U-Net based discriminator for our UDC image generation task. Compared to other discriminators that classify the input image x into being real and fake, our U-Net-like discriminator additionally performs this classification on a per-pixel basis, segmenting image x into real and fake regions, along with the whole image classification of x from the encoder part of the discriminator. Thus, the discriminator loss is now can be computed by taking the decisions from both encoder D^U_{enc} and decoder D^U_{dec}:

$$\mathcal{L}_{D^U} = \mathcal{L}_{D^U_{enc}} + \mathcal{L}_{D^U_{dec}}. \tag{1}$$

The $\mathcal{L}_{D^U_{enc}}$ follows the vanilla adversarial loss

$$\mathcal{L}_D = -\mathbb{E}_x[logD^U_{enc}(x)] - \mathbb{E}_z[log(1 - D^U_{enc}(G(z)))], \tag{2}$$

and the loss for the decoder $\mathcal{L}_{D^U_{dec}}$ is computed as the LSGAN [1] loss over all pixels:

$$\begin{aligned}\mathcal{L}_{D^U_{dec}} = &\frac{1}{2}\mathbb{E}_x\left[(D(x) - b)^2\right] \\ &+ \frac{1}{2}\mathbb{E}_z\left[(D(G(z)) - a)^2\right],\end{aligned} \tag{3}$$

where $a = 1$ and $b = 0$. Correspondingly, the generator objective becomes:

$$\mathcal{L}_G = -\mathbb{E}_z[logD^U_{enc}(G(z))] - \frac{1}{2}\mathbb{E}_z\left[(D(G(z)) - c)^2\right], \tag{4}$$

where $c = 1$. Using the powerful U-Net-like discriminator D^U, we can encourage the generator to focus on both global structures and local details, and thus synthesizing realistic images.

Our U-Net-like discriminator consists of four encoder stages, four decoder stages, and a bottleneck stage. Each encoder stage contains two convolutions and one down-sampling layer, a 2×2 max pooling layer with a stride size of two. The bottleneck stage has three convolutions. And in the middle of the bottleneck

Table 1. The qualitative results of various restoration models trained under the same training setting. DWFormer$^+$ is trained using generated and real datasets.

Methods	P-OLED		T-OLED		Overhead	
	PSNR	SSIM	PSNR	SSIM	Params	MACs
UNet [2]	29.31	0.934	36.60	0.971	8.94M	17.09G
SGDAN [3]	30.80	0.947	36.62	0.971	21.1M	7.25G
RDUNet [4]	30.02	0.941	38.16	0.980	47.93M	46.01G
ResUNet [4]	30.54	0.945	37.84	0.979	16.50M	15.79G
PDCRN [5]	31.04	0.958	37.83	0.978	3.65M	6.31G
DRANet [6]	31.86	0.949	38.84	0.983	79.01M	168.98G
DWFormer(Ours)	33.21	0.960	38.96	0.984	1.21M	13.46G
DWFormer$^+$(Ours)	34.22	0.964	39.55	0.986	1.21M	13.46G

stage, we additionally use a global average pooling with a multi-layer perception (MLP) to derive the feature vector classified as real or fake. The decoder stage consists of two convolutions and one deconvolution up-sampling layer. And the end of the decoder, we use a pixel-wise convolution to produce the feature map, which has the same resolution as the original image but only one dimension for pixel-wise classification.

B Models Comparison Under the Same Training Setup

Our work is based on the UDC 2020 Image Restoration Challenge, and the report [3] shows the results of the competition and some models with promising performance. However, these models are trained under different training settings, and some used tricks, thus making unfair comparisons. Therefore, we try to reproduce these models and compare them under the same training settings without any tricks. Specifically, we crop both P-OLED and T-OLED datasets [2] provided by UDC 2020 Image Restoration Challenge into patches of 256×256. And we set the batch size to 64 and randomly used flips and rotations for data augmentation. All models are trained with Adam optimize ($\beta_1 = 0.9$, and $\beta_2 = 0.999$) for 300 epochs. The initial learning rate is set to 2×10^{-4} and the cosine annealing strategy is adopted to steadily decrease the learning rate to 2×10^{-6}. We show the performance comparison results in Table 1. It can be seen that some models have a significant performance drop, while our model still has promising performance, indicating that our model converges faster. Further, we experimentally show that the performance of our model is still optimal while ensuring equal training overhead (the training time is the same, but the training epoch varies for different models due to different computation costs).

Table 2. Quantitative Evaluation On UDC Benchmark

Methods	P-OLED		T-OLED	
	PSNR	SSIM	PSNR	SSIM
Real	32.95	0.948	38.10	0.975
SGM	27.92	0.899	34.21	0.943
SGM + Real	31.34	0.931	37.54	0.971
U-Net	30.13	0.929	35.27	0.951
MPGNet	30.80	0.928	36.26	0.962
MPGNet + Real	34.02	0.955	39.16	0.985

C Effectiveness of MPGNet-generated Datasets

Since we use a pre-trained restoration model in the generation process, we perform the same experiments with other restoration models to verify the effectiveness of the generated data. Here, we use PDCRN and RDU-Net with promising performance on P-track and T-track, respectively, as restoration models. We first use SGM-generated and MPGNet-generated datasets to train the restoration models and evaluate them on the UDC benchmark. We also train the restoration model with the real dataset for comparison and use the real and generated data together to train the restoration model. The result shown in Table 2 illustrates the quantitative results. Our method outperforms SGM by 2.88 dB on the P-OLED track and 2.05 dB on the T-OLED track by only using the synthetic dataset, demonstrating the effectiveness of MPGNet. And when we use the generated data together with the real data as training data, the restoration models outperform the models trained with the real data only by 1.07 dB on the P-OLED track and 1.06 dB on the T-OLED track. From the experimental results, we can see that the data generated by MPGNet are much better than those generated by SGM and are closer to the real ones. Moreover, our generated datasets are suitable for many different restoration models, indicating the diversity and well generalization of our generated datasets.

D Different Batchsize on a Single GPU

Considering that our restoration model uses BatchNorm for normalization, thus the size of the batch size during training will impact the performance, and we will make an exploration here. To guarantee the consistency of the experiments and utilize the asynchronous BN property of PyTorch, we trained part of the experiments again with an 8-card RTX 3080 GPU to compare the impact of different batch sizes on a single GPU with the same experimental setup. Note that we train our original model on a 4-card RTX 3090 GPU.

Table 3. Performance comparison of DWFormer's modules' effectiveness on normalization batch size (NBS).

Methods	P-OLED		T-OLED		MACs
	NBS=16	NBS=8	NBS=16	NBS=8	
DWFormer	33.21	33.17	38.96	38.90	13.46G
ACA → None	32.62	32.47	38.45	38.35	13.42G
ACA → SE	33.00	32.89	38.72	38.63	13.43G
DWB → Swin	33.07	32.98	38.84	38.72	16.22G
BN → LN	33.11	33.10	39.90	38.90	13.46G

E Inference Time

Table 4. Runtime comparison of the models

Method	Hardware	Resolution			
		256×256	512×512	1024×1024	1024×2048
UNet	RTX 2070	175.51	46.97	12.29	6.21
	RTX 3080	369.55	105.38	26.65	13.40
SGDAN	RTX 2070	327.21	107.24	30.03	15.30
	RTX 3080	486.38	233.15	58.92	30.0
RDUNet	RTX 2070	69.69	19.74	5.21	2.62
	RTX 3080	147.91	45.23	11.43	5.86
ResUNet	RTX 2070	194.50	55.39	14.10	7.11
	RTX 3080	362.40	117.01	31.52	15.31
PDCRN	RTX 2070	202.71	97.77	27.54	14.02
	RTX 3080	215.88	139.41	51.13	25.89
DRANet	RTX 2070	11.90	3.76	0.94	0.48
	RTX 3080	58.52	16.65	4.29	2.15
DWFormer	RTX 2070	145.77	72.90	18.74	9.42
	RTX 3080	160.73	129.41	35.04	17.92

Considering that UDC is a technology applied in edge devices, thus the inference time of the restoration models is very important. Since the inference time is highly correlated with the GPU and its running state, we perform the following experimental setup to avoid coincidence. We run five epochs for each model. Each epoch contains 100 identical images to get five average inference

times and choose the median of the results as our final inference time. We conducted the inference time of our replicated models on a single NVIDIA 2070 and 3080 GPU, and we show the results in Table 4. From the results, we can see that our DWFormer can restore the image of size 1024×2048 less than 0.1s on both NVIDIA 2070 and 3080 GPU, locating in the third rank of all models. Although our model's inference speed is not the fastest, our model has promising results on both the P-OLED track and the T-OLED track, so only one model needs to be deployed while using different parameters. Therefore, our model achieves a balance between performance and practical deployment.

F MPGNet Composition Exploration

Considering that the quantization noise of StarLight [7] is fixed during image generation, we explore the impact of signal-dependent quantization noise and the location of quantization noise (see Table 5).

Table 5. Ablation study of quantization noise type and location. Dep means we utilize input to estimate the quantization noise distribution parameters, and LA means we add the quantization noise after the signal-dependent and signal-independent noise.

Methods	P-OLED		T-OLED	
	PSNR	SSIM	PSNR	SSIM
MPGNet	31.92	0.940	37.31	0.969
→ Dep	$31.94_{\uparrow 0.02}$	$0.940_{\uparrow 0.000}$	$37.32_{\uparrow 0.01}$	$0.970_{\uparrow 0.001}$
→ LA	$31.91_{\downarrow 0.01}$	$0.940_{\downarrow 0.000}$	$37.29_{\downarrow 0.02}$	$0.968_{\downarrow 0.001}$
→ Dep and LA	$31.95_{\uparrow 0.03}$	$0.941_{\uparrow 0.001}$	$37.33_{\uparrow 0.02}$	$0.970_{\uparrow 0.001}$

Also, we replace our MPGNet's module with BNCR-GAN's [8] blur and noise modules to build generators (see Table 6). Results show that our MPGNet performs best, which indicates the effectiveness of our noise and blur module.

Table 6. Ablation study of different types of degradation modules. We use blur and noise in BNCR-GAN to build generators.

Methods	P-OLED		T-OLED	
	PSNR	SSIM	PSNR	SSIM
MPGNet	31.92	0.940	37.31	0.969
w/ BNCR-GAN Noise	$31.30_{\downarrow 0.62}$	$0.934_{\downarrow 0.006}$	$36.66_{\downarrow 0.65}$	$0.962_{\downarrow 0.007}$
w/ BNCR-GAN Blur	$31.01_{\downarrow 0.91}$	$0.931_{\downarrow 0.009}$	$36.61_{\downarrow 0.70}$	$0.962_{\downarrow 0.007}$

References

1. Mao, X., Li, Q., Xie, H., Lau, R.Y., Wang, Z., Paul Smolley, S.: Least squares generative adversarial networks. In: ICCV. (2017) 2794–2802 1
2. Zhou, Y., Ren, D., Emerton, N., Lim, S., Large, T.: Image restoration for under-display camera. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 9179–9188 2
3. Zhou, Y., Kwan, M., Tolentino, K., Emerton, N., Lim, S., Large, T., Fu, L., Pan, Z., Li, B., Yang, Q., et al.: Udc 2020 challenge on image restoration of under-display camera: Methods and results. In: European Conference on Computer Vision, Springer (2020) 337–351 2
4. Yang, Q., Liu, Y., Tang, J., Ku, T.: Residual and dense unet for under-display camera restoration. In: European Conference on Computer Vision, Springer (2020) 398–408 2
5. Panikkasseril Sethumadhavan, H., Puthussery, D., Kuriakose, M., Charangatt Victor, J.: Transform domain pyramidal dilated convolution networks for restoration of under display camera images. In: European Conference on Computer Vision, Springer (2020) 364–378 2
6. Nie, S., Ma, C., Chen, D., Yin, S., Wang, H., Jiao, L., Liu, F.: A dual residual network with channel attention for image restoration. In: European Conference on Computer Vision, Springer (2020) 352–363 2
7. Monakhova, K., Richter, S.R., Waller, L., Koltun, V.: Dancing under the stars: video denoising in starlight. In: CVPR. (2022) 5
8. Kaneko, T., Harada, T.: Blur, noise, and compression robust generative adversarial networks. In: CVPR. (2021) 5