ReBaIR: Reference-Based Image Restoration

Supplementary Material

1. Training Details

All our methods were implemented and trained using PyTorch [15]. Training was performed on Nvidia RTX4090 GPUs while evaluation, including runtime analysis, was performed on Nvidia RTX A6000 GPUs. All our training was performed on the LMR [25] dataset. Both ground truth images and reference images were randomly cropped to size 256×256. The LMR training dataset provides 5 reference images for each ground truth images. During training we vary the number of reference images for each batch. The number of reference images is chosen randomly between 1 and 5. Throughout all training we use the AdamW optimizer with an initial learning rate of $1e-4$ and we use cosine annealing to decay the learning rate to $1e-5$. Our method is trained in two steps.

In the first step the two refinement stages are trained independently each for a total of 300k steps. For this step we use use a batch size of 4 and a single GPU for SR tasks (SRx4, RealSRx4, ASR). For other tasks (Denoise, JPEG) we use a batch size of 2 and 2 GPUs. The reason for this is the increased backbone memory consumption for non-SR tasks as the input is higher resolution.

In the second step we train the two refinement stages jointly for a total of 150k steps. For this step we use a batch size of 2 and 2 GPUs for all models.

An overview of training times for each model is provided in Table 7. Here, we report training times for both steps and the total training time required. Note that since stages 1 & 2 can be trained in parallel the total training time is the maximum time required for stage 1 & 2 plus the combined training time.

Task	Backbone	Stage 1	Stage 2	Combined	Total
SRx4	DRCT	35h	41h	19h	60h
SRx4	SwinIR	31h	37h	17h	54h
RealSRx4	SwinIR	32h	39h	31h	70h
ASR	LIIF RDN	33h	40h	19h	59h
Denoise	Restormer	25h	31h	22h	53h
JPEG	SwinIR	66h	80h	39h	119h

Table 7. Overview of training times for different tasks and backbones. Since stages 1 & 2 can be trained in parallel the total training time is the maximum time required for stage 1 & 2 plus the combined training time.

2. Numeric Evaluation

Throughout our whole numeric evaluation presented in Tables 1&2 we compute PSNR and SSIM on the Y component in the YCbCr color space while LPIPS is computed on RGB images. To avoid boundary artifacts we ignore a margin of 16px around the image when computing each metric.

3. Additional Ablation

Confidence	Sub-Pixel Offset	PSNR ↑	SSIM ↑	LPIPS ↓
✓	✓	32.44	0.916	0.101
✗	✓	32.41	0.915	0.101
✓	✗	32.43	0.916	0.102

Table 8. Ablation study showing the effect of utilizing confidence and sub-pixel offsets provided by PDCNet+ in our refinement stage. Both have a minor positive effect.

In addition to the ablation study performed in the main paper we further ablate our second refinement stage where PDCNet+ is used to compute correspondences. Specifically, we investigate the benefits of confidence values and sub-pixel accuracy provided by PDCNet+. The results are shown in Table 8. We see that both confidence values and sub-pixel offsets have a minor positive influence on output quality.

4. Architecture Details

In this section we describe each part of our architecture in more detail.

RDB Residual Dense Blocks are parametrized by the number of input channels C, the growth rate G, and the number of layers N. Unless explicitly mentioned otherwise we use $N = 4$ and $G = \frac{C}{4}$ throughout our method. Also, we use the *LeakyReLU* activation function with a negative slope of 0.1.

MRFR MHA Our Multi-Reference Feature Refinement module employs multi-head-attention to aggregate information from multiple aligned reference feature maps into a unified reference feature map. Note that this operation is performed at each scale separately and we refer to the the number of feature dimensions at a given scale as C. Our mha module uses $\frac{C}{32}$ attention heads, each of size 32. Queries and key-value pairs are extracted using two separate RDBs. The attention mechanism then operates on each spacial location independently.

Feature Warping Feature warping is generally performed using nearest neighbor interpolation. In the second

refinement stage, where mappings between base and reference images are computed using PDCNet+, *sub-pixel offsets* and *confidence* are added along the channel dimension to the warped feature map. By *sub-pixel offsets* we refer to the 2D vector pointing from the location the mapping indicates to the nearest discrete pixel location. *Confidence* is a single number between 0 and 1 provided by PDCNet+ and it indicates the confidence PDCNet+ has in the mapping being correct at this location.