A. Appendix

In this document, we first provide further derivation of the proposed method in Sec. A.1. Then we demonstrate the way to initialize the wavelet and the low memory usage of our method in practical applications in Sec. A.2 and Sec. A.3, respectively. Finally, we show more visual comparisons between the proposed method and state-of-the-art methods in Sec. A.4.

A.1. Further Derivation on the Proposed Method

For the $z$-transform $X(z)$ of the input $x$ in Sec. ??, which undergoes a forward wavelet transform to produce high and low frequency components with a downsampling factor of 2 can be expressed as,

$$X_{\text{down,low}}(z) = \frac{1}{2} \left[ X(z^{\frac{1}{2}}) A_0(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}}) A_0(-z^{\frac{1}{2}}) \right]$$

(12)

$$X_{\text{down,high}}(z) = \frac{1}{2} \left[ X(z^{\frac{1}{2}}) A_1(z^{\frac{1}{2}}) + X(-z^{\frac{1}{2}}) A_1(-z^{\frac{1}{2}}) \right]$$

(13)

After upsampling by a factor of 2, the $z$-transform of the reconstructed $x$ is obtained as,

$$X_{\text{up}}(z) = \frac{1}{2} [X(z) A_0(z) + X(-z) A_0(-z)] S_0(z)$$

$$\quad \quad + \frac{1}{2} [X(z) A_1(z) + X(-z) A_1(-z)] S_1(z)$$

$$= \frac{1}{2} [A_0(z) S_0(z) + A_1(z) S_1(z)] X(z)$$

$$\quad \quad + \frac{1}{2} [A_0(-z) S_0(-z) + A_1(-z) S_1(-z)] X(-z)$$

(14)

where $X(-z)$ denotes an aliased signal in the reconstruction, which is set to cancel its effect,

$$A_0(-z) S_0(z) + A_1(-z) S_1(z) = 0$$

(15)

Obviously, that gives another item,

$$A_0(z) S_0(z) + A_1(z) S_1(z) = 2$$

(16)

Splicing Eq. 15 and Eq. 16 gives the perfect reconstruction condition. Using the convolutional property of the $z$-transform, the loss function shown in $L_{\text{wavelet}}$ (see Sec. ??) optimized toward the zeros can be obtained.

A.2. Initialization of Wavelet Convolution

For $K_{\text{wavelet}}$ in Eq. ??, we performed the following experiments using db2 wavelet initialization [10], haar wavelet initialization [3] and random initialization.

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR</th>
<th>SSIM</th>
<th>wavelet kernel size</th>
</tr>
</thead>
<tbody>
<tr>
<td>db2 [10]</td>
<td>32.69</td>
<td>0.931</td>
<td>4×4</td>
</tr>
<tr>
<td>haar [3]</td>
<td>32.62</td>
<td>0.931</td>
<td>2×2</td>
</tr>
<tr>
<td>random</td>
<td>32.53</td>
<td>0.930</td>
<td>2×2</td>
</tr>
</tbody>
</table>

Table 7. Comparison of results of different initialization wavelet ways.

As shown in Tab. 7, better accuracy can be obtained by using subtly constructed classical wavelets for initialization, which corresponds to the introduction of artificial prior knowledge. Considering the balance between efficiency and accuracy, in the specific implementation of MLWNet, we use haar wavelets for initialization. From the above tables, it can be seen that the initialization using haar wavelet with a priori information results in a higher PSNR metric of 0.09 dB compared to random initialization.

A.3. GPU Memory Usage under High-Resolution Images

In this section, we use a Tesla A40 with 48GB GPU memory to test the average memory usage on the RSBlur dataset (the image resolution is approximately 1920×1200). The
comparison results with various advanced algorithms are shown in Tab. 8. It is noted that our method uses only 1.4GB more memory than MIMO-UNet+ [1], and is only one-fifth of the advanced algorithm FFTformer [5]. Combining Tab. ?, Tab. ?? and Tab. ??, it can be seen that our proposed MLWNet is able to well balance the accuracy and model complexity at large resolution, and the smaller memory allocation also ensures that the algorithm can be deployed on some GPUs with poorer arithmetic power.

A.4. More Experimental Results

In this section, we provide more visual comparisons of the proposed method and state-of-the-art ones on RealBlur-J [8] (see Fig. 9, Fig. 10) and RSBlur [9] (see Fig. 11, Fig. 12, Fig. 13) datasets.

References


<table>
<thead>
<tr>
<th>Method</th>
<th>MIMO-UNet+</th>
<th>MSSNet</th>
<th>MPRNet</th>
<th>Stripformer</th>
<th>GRL</th>
<th>FFTformer</th>
<th>MLWNet-B(Ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU memory(GB)</td>
<td>8.5</td>
<td>13.7</td>
<td>21.8</td>
<td>27.5</td>
<td>Out of memory</td>
<td>47.3</td>
<td>9.9</td>
</tr>
</tbody>
</table>

Table 8. Memory comparisons of our method and the advanced algorithms. “GPU memory” denotes the maximum GPU memory consumption that is computed by the “torch.cuda.max_memory_allocated()” function.

Figure 9. Visualization results for the RealBlur-J [8] dataset. Our proposed MLWNet performs excellently under the conditions of rich light, shadow, and details.

Figure 10. Visualization results for the RealBlur-J [8] dataset. Our proposed MLWNet restores the image closest to GT, and it can be seen that MLWNet excels in terms of color, sharpening, and details.
Figure 11. Visualization results for the RSBlur [9] dataset. Our proposed MLWNet restores the sharpest leg details of fast-moving pedestrians, which shows that MLWNet still performs well in the difficult motion blur restoration.

Figure 12. Visualization results for the RSBlur [9] dataset. Our proposed MLWNet restores the most realistic image in terms of light and shadow. It can be easily seen that MLWNet has the least retention on residual shadows.
Figure 13. Visualization results for the RSBlur [9] dataset. Our proposed MLWNet restores the clearest text details and achieves eye-pleasing results in noise suppression.