Intriguing Findings of Frequency Selection for Image Deblurring

Xintian Mao[1], Yiming Liu[1], Fengze Liu[2], Qingli Li[1], Wei Shen[3], Yan Wang[1]*

[1]Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University
[2]Department of Computer Science, the Johns Hopkins University
[3]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
{52265904010,52205904009}@stu.ecnu.edu.cn, fliu23@jhu.edu, qlli@cs.ecnu.edu.cn, wei.shen@sjtu.edu.cn, ywang@cee.ecnu.edu.cn

Abstract

Blur was naturally analyzed in the frequency domain, by estimating the latent sharp image and the blur kernel given a blurry image. Recent progress on image deblurring always designs end-to-end architectures and aims at learning the difference between blurry and sharp image pairs from pixel-level, which inevitably overlooks the importance of blur kernels. This paper reveals an intriguing phenomenon that simply applying *ReLU* operation on the *frequency* domain of a blur image followed by inverse Fourier transform, *i.e.*, frequency selection, provides faithful information about the *blur pattern* (*e.g.*, the blur direction and blur level, implicitly shows the kernel pattern). Based on this observation, we attempt to leverage kernel-level information for image deblurring networks by inserting Fourier transform, ReLU operation, and inverse Fourier transform to the standard ResBlock. 1×1 convolution is further added to let the network modulate flexible thresholds for frequency selection. We term our newly built block as Res FFT-ReLU Block, which takes advantages of both kernel-level and pixel-level features via learning frequency-spatial dual-domain representations. Extensive experiments are conducted to acquire a thorough analysis on the insights of the method. Moreover, after plugging the proposed block into NAFNet, we can achieve 33.85 dB in PSNR on GoPro dataset. Our method noticeably improves backbone architectures without introducing many parameters, while maintaining low computational complexity. Code is available at https://github.com/DeepMed-Lab/DeepRFT-AAAI2023.

Introduction

Image deblurring aims at removing blurring artifacts to recover sharp images (Cho et al. 2021). The blurring of an image can be caused by many factors, *e.g.*, camera shake, objects movement, out-of-focus optics, etc. The blurry image leads to visually low quality and hampers subsequent high-level vision tasks, ranging from security, medical imaging to object recognition (Chen et al. 2020).

Image deblurring by frequency domain operations was very popular decades ago, based on a simple assumption that image blur may be due to the Point Spread Function (PSF) of the sensor, sensor motion, and other reasons (Banham and Katsaggelos 1997). Thus, the motion blurred image is expressed by convolving a latent sharp image with the PSF, which can be most easily instantiated in the frequency domain (Chakrabarti, Zickler, and Freeman 2010; Xu, Zheng, and Jia 2013; Hu et al. 2014; Pan et al. 2016). Here we take a simple example and assume that the blur kernel does not vary spatially: $B(\omega) = F(\omega)G(\omega) + N(\omega)$, where $F(\omega)$, $G(\omega)$, $B(\omega)$ and $N(\omega)$ are Fourier transforms of the sharp image, the blur kernel (PSF), the blurry image and the sensor noise, respectively, and $\omega \in [-\pi, \pi]^2$. This gives us an important cue that frequency domain offers us abundant information on image deblurring tasks which should not be overlooked.

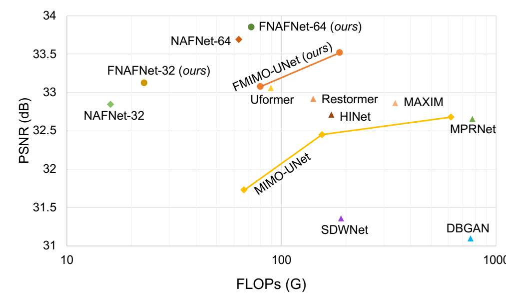

Figure 1: PNSR vs. computational cost on the GoPro dataset (Nah, Kim, and Lee 2017). Our method performs much better than baseline methods: MIMO-UNet (Cho et al. 2021) and NAFNet (Chen et al. 2022), and other state-of-the-arts.

Deep networks are popular for their end-to-end learning ability. The field of image deblurring has made significant advances riding on the wave of deep networks. DeepDeblur (Nah, Kim, and Lee 2017), pioneers the technique of end-to-end trainable methods, directly mapping a blurry image to its paired sharp image by a Convolutional Neural Network (CNN). It designs a multi-scale architecture, and uses a modified residual network structure (He et al. 2016) called *ResBlock* (see Fig. 4 (a)) to focus on learning the difference between blurry and sharp image pairs. Thereafter, end-to-end learning strategy with ResBlock is proven to be effective in image deblurring, and becomes a mainstream approach in recent years (Tao et al. 2018; Zhang et al. 2019; Gao et al. 2019; Park et al. 2020; Yuan, Su, and Ma 2020; Zamir et al.

*Corresponding Author: ywang@cee.ecnu.edu.cn.

2021; Chen et al. 2021; Zou et al. 2021; Purohit et al. 2021; Cho et al. 2021; Chen et al. 2022). But, these methods overlook the importance of blur kernels.

In this paper, we reveal an intriguing phenomenon that taking the inverse Fourier transform on *frequency selection* (*e.g.*, ReLU on the frequency domain) of a blurry image acts as learning *blur pattern* from the blurry image, indicating the blur direction and blur level, implicitly showing the *kernel* pattern. The faithful blur kernel information provided by such operations motivates us to insert Fourier transform, ReLU, and inverse Fourier transform to the standard ResBlock to take advantages of both *kernel*-level and *pixel*-level features via fusing frequency-spatial dual-domain representations. Furthermore, we investigate ReLU on the frequency domain from a new perspective, and experiments show that setting different thresholds instead of $0 + j0$ ($0 + j0$ is for ReLU) for frequency selection give different deblurring results. With this new viewpoint, we find that adding convolution after Fourier transform helps the network modulate flexible thresholds for selecting frequencies and can further promote image deblurring performance. To sum up, we propose a new, efficient and *plug-and-play* ResBlock, termed as Residual (Res) Fast Fourier Transform (FFT)-ReLU Block, to replace standard ResBlock. Our Res FFT-ReLU Block inserts a FFT-ReLU stream, consisting of 4 simple operations: 2D real FFT, 1×1 convolutions, ReLU and inverse 2D real FFT, into ResBlock.

The effectiveness of the Res FFT-ReLU Block is compared and verified by plugging in different architectures on three datasets: GoPro (Nah, Kim, and Lee 2017), HIDE (Shen et al. 2019) and RealBlur (Rim et al. 2020) datasets. Substantial ablation studies are conducted to explore insights of the FFT-ReLU stream. It is worth mentioning that after plugging the proposed stream to MIMO-UNet+ (Cho et al. 2021) and NAFNet (Chen et al. 2022), the new models, which are termed as FMIMO-UNet+ and FNAFNet, can achieve 33.52 dB and 33.85 dB respectively in terms of PSNR on GoPro dataset. Our method noticeably improves backbone architectures without introducing too many parameters, while maintaining low computational complexity. The PSNR vs. FLOPs (G) compared with state-of-the-art methods are shown in Fig. 1.

Related Works

Deep Image Deblurring Deep learning methods have achieved significant success in image deblurring (Sun et al. 2015; Nah, Kim, and Lee 2017) as well as other low-level vision tasks such as image denoise (Cheng et al. 2021; Zamir et al. 2020), image deraining (Jiang et al. 2020) and image super-resolution (Mei, Fan, and Zhou 2021; Dong et al. 2016; Zhang et al. 2018b; Guo et al. 2020). Sun *et al.* (Sun et al. 2015) propose to estimate the spatially-varying kernels of motion blur by a CNN. But, since the characteristics of blur are complex, the blur kernel estimation method is not practical in real scenarios. Later, DeepDeblur (Nah, Kim, and Lee 2017) directly maps a blurry image to its sharp counterpart. Scale-recurrent network (Tao et al. 2018) proposes an encoder-decoder structure to yield training feasibility. Adversarial training(Kupyn et al. 2018, 2019; Zhang et al. 2020) and Recurrent Neural Networks(Zhang et al. 2018a; Park et al. 2020) also have been extensively studied. Most of these networks perform CNNs on the spatial domain to recover the sharp image. MAXIM (Tu et al. 2022) proposes MLP-based building blocks, which requires a big batch size for training. NAFNet (Chen et al. 2022) designs computationally efficient networks from the baseline, which even achieves 33.69 dB PSNR on GoPro. Instead of designing a brand-new end-to-end image deblurring architecture, we reveal an intriguing phenomenon on frequency selection, and shed new light on improving the deblurring performance by incorporating faithful information about the blur pattern.

Transformer/non-local has strong global context modeling ability and has shown its great promise in various computer vision tasks. Some transformer-based image restoration methods have been proposed, such as SwinIR (Liang et al. 2021), Restormer (Zamir et al. 2022) and Uformer (Wang et al. 2022). But the considerable computational complexity usually hampers their usage in efficient image restoration. We test the model of SwinIR (Liang et al. 2021) and Restormer (32.92 dB on GoPro) (Zamir et al. 2022) on GoPro dataset, which take 1.99s and 1.14s per image, respectively, even much slower than MPRNet, while our FMIMO-UNet (33.08 dB) takes 0.339s per image.

End-to-end Deblur Model with ResBlock DeepDeblur (Nah, Kim, and Lee 2017) designs a residual block (ResBlock) based on Conv-ReLU-Conv structure. Thereafter, ResBlock has become one fundamental block in image deblurring (Tao et al. 2018; Zhang et al. 2019; Park et al. 2020; Purohit and Rajagopalan 2020; Cho et al. 2021). Various efforts have been devoted to modifying the ResBlock, *e.g.*, the content-aware processing module proposed by SAPHN (Suin, Purohit, and Rajagopalan 2020), the channel attention block proposed by MPRNet (Zamir et al. 2021), the HIN block proposed by HINet (Chen et al. 2021), and the dilated conv block proposed by SDWNet (Zou et al. 2021).

Applications of Fourier Transform In recent years, some methods extract information from the frequency domain to fulfill different tasks (Chi et al. 2019; Rippel, Snoek, and Adams 2015; Zhong et al. 2018; Yang and Soatto 2020; Rao et al. 2021; Suvorov et al. 2022). FDA (Yang and Soatto 2020) swaps the low-frequency spectrum between images to mitigate the influence caused by the images' style change for image segmentation. GFNet (Rao et al. 2021) learns long-term spatial dependencies in the frequency domain for image classification. LaMa (Suvorov et al. 2022) applies the structure of fast Fourier convolution (Chi, Jiang, and Mu 2020) to image inpainting. In image deblurring, SDWNet (Zou et al. 2021) introduces wavelet transform into deep networks. In this paper, we reveal an intriguing phenomenon of frequency selection for image deblurring.

Method

Empirical Findings of Frequency Selection via ReLU Operation

We will describe the main observation insights of this paper in this section. The blurry image can be modeled as

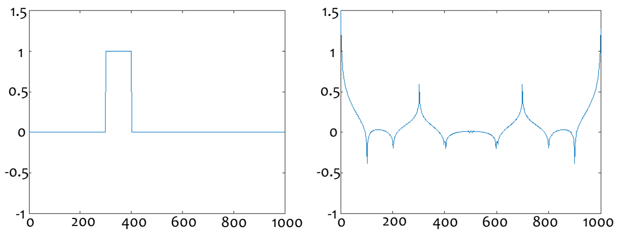

Figure 2: Given a top-hat function f on the left, its $\mathcal{F}^{-1}(|\mathcal{F}(f)|$ is plotted on the right, with obvious peaks, where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ mean discrete Fourier and inverse Fourier transform.

convolving the latent sharp image with the blur kernel. For simplicity, we elaborate the deviation in 1D scenario. The "unknown" sharp image is defined as $f(t) = \begin{cases} 1 & |t| < \tau \\ 0 & |t| > \tau \end{cases}$ The blur kernel of a simple motion blur can be denoted as $g(t) = \delta(t) + \delta(t+\epsilon)$, where $\delta(\cdot)$ is the Direct delta function, and ϵ is a very small value.

Remark 1: Based on the above assumption, let's define a blurry image $b = g \otimes f$, where $\otimes$ means convolution, and we drop t for simplicity. Taking the inverse Fourier transform (*i.e.*, $\mathcal{F}^{-1}(\cdot)$) after ReLU (*i.e.*, $\sigma(\cdot)$) on the blurry image in the frequency domain ($\mathcal{F}(\cdot)$ means Fourier transform) is written as $\hat{b} = \mathcal{F}^{-1}(\sigma(\mathcal{F}(b)))$, which can separate blur pattern component from other components. ReLU is applied to the real and imaginary parts respectively.

More details to support **Remark 1** are given in the supplementary material. One can easily decompose $\hat{b}$ and obtain a separate component from $\hat{b}$ as $(\delta(t) + 1/2(\delta(t-\epsilon) + \delta(t+\epsilon))) \otimes \mathcal{F}^{-1}(|\mathcal{F}(f(t))|)$.

We observe that an absolute operation on Fourier transform of a top-hat function will produce peaks after inverse Fourier transform, as examples shown in Fig. 2. We plot the discrete Fourier transform. Based on the phenomenon revealed in Fig. 2, $\mathcal{F}^{-1}(\sigma(\mathcal{F}(g \otimes f)))$ will separate a component containing the convolution of g and some peak values, when f is a top-hat function.

Without loss of generality, we can extend **Remark 1** to more complicated scenarios. Fig. 3 shows examples of feature selection derived from a clean and blurry images caused by blur kernels with different blur direction, blur levels and mixtures of blur. As may be observed, taking inverse Fourier transform on frequency selection via ReLU acts as a sort of learning blur pattern directly from the blurry image.

Simple Fourier Transform with ReLU Stream

Basic Building Block for Image Deblurring The residual building block in image deblurring tasks is called ResBlock. Specifically, standard ResBlock learns pixel-level features, consisting of two 3×3 convolutional layers and one ReLU layer in between, as shown in Fig. 4 (a). Using inverse Fourier transform after selecting frequency from a blurry image $\mathcal{F}^{-1}(\sigma(\mathcal{F}(b)))$ generates blur patterns, implicitly showing the kernel pattern (see Eq. 3 in supplementary material and Fig. 3, $\mathcal{F}^{-1}(\sigma(\mathcal{F}(b)))$ can be considered as a linear combination of a blur pattern image $\mathcal{F}^{-1}(|\mathcal{F}(b)|)$ and b). If such operations can be inserted into an end-to-end image deblurring network, the network will be able to learn both kernel-level and pixel-level information. Many methods can be applied to fuse kernel-level blur pattern features generated by $\mathcal{F}^{-1}(\sigma(\mathcal{F}(b)))$ and the pixel-level features. Instead of designing a complicated fusion block, we simply replace the *identity* mapping by $\mathcal{F}^{-1}(\sigma(\mathcal{F}(\mathbf{Z})))$ for the sake of light computation, *i.e.*, replace the *identity* mapping in Fig. 4(a) by the left most stream, termed as simple-FFT-ReLU stream in Fig. 4(b). The position of simple-FFT-ReLU stream *w.r.t.* the standard ResBlock (see Fig. 5(a)) can be changed, and our experiments show that fusing kernel-level and pixel-level features either in parallel or sequentially can boost the results (see Table 1).

Analysis of ReLU on Frequency Domain from A New Perspective We analyze ReLU in frequency domain from another perspective. The phase and amplitude of a complex number $z = me^{j\beta}$ are $e^{j\beta}$ and $m \geq 0$ respectively. A Fourier transformed feature gives the phase and amplitude components. Applying ReLU (denoted as $\sigma(\cdot)$) on the frequency domain of a feature is defined as $\sigma(me^{j\beta}) = m\sigma(e^{j\beta})$. This is considered as applying ReLU on the phase of the feature. As suggested by Pan (Pan et al. 2019), the phase of a blurry image plays an important role for deblurring, providing faithful information about the motion pattern. Applying ReLU indicates setting $T = 0 + j0$ as the selective threshold. What will happen if we change the threshold for frequency selection? As shown in Table 2 in the experiment, setting different thresholds on ReLU, *i.e.*, $m\sigma(e^{j\beta} - \frac{T}{m}) + T$, gives different results, *e.g.*, $T = 100(1 + j)$ leads to a better result compared with original ReLU. A proper threshold for selecting frequency is important for deblurring. But T has to be set manually. To let the network modulate flexible thresholds for selecting frequencies, convolution can be further added after Fourier transform. For the sake of simplicity, let $a \cdot ke^{j\beta} + b$ denote the feature after 1×1 conv in frequency, where a and b are learnable complex values. $m\sigma(e^{j\beta} - \frac{T}{m}) + T$ is then replaced by $m\sigma(a \cdot e^{j\beta} + \frac{b}{m})$ (ReLU after convolution).

Res FFT-ReLU Block

From the above analysis, we propose a Residual Fast Fourier Transform with ReLU Block (Res FFT-ReLU Block) to replace the widely-used ResBlock. As shown in Fig. 4(c), different from Fig. 4(b), we keep the *identity* mapping which assists network training.

As shown in Fig. 4(c), besides a normal spatial residual stream, we simply add another stream based on a channel-wise FFT (Brigham and Morrow 1967). DFT is widely used in modern signal processing algorithms, whose 1D version can be derived by $X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn}$, where $x[n]$ is a sequence of N complex numbers, $X[k]$ indicates the spectrum at the frequency $\omega_k = 2\pi k/N$, and j represent the imaginary unit. It is clear that the spectrum at any frequency has global information. Noted that the DFT of a real signal $x[n]$ is *conjugate symmetric*, *i.e.*

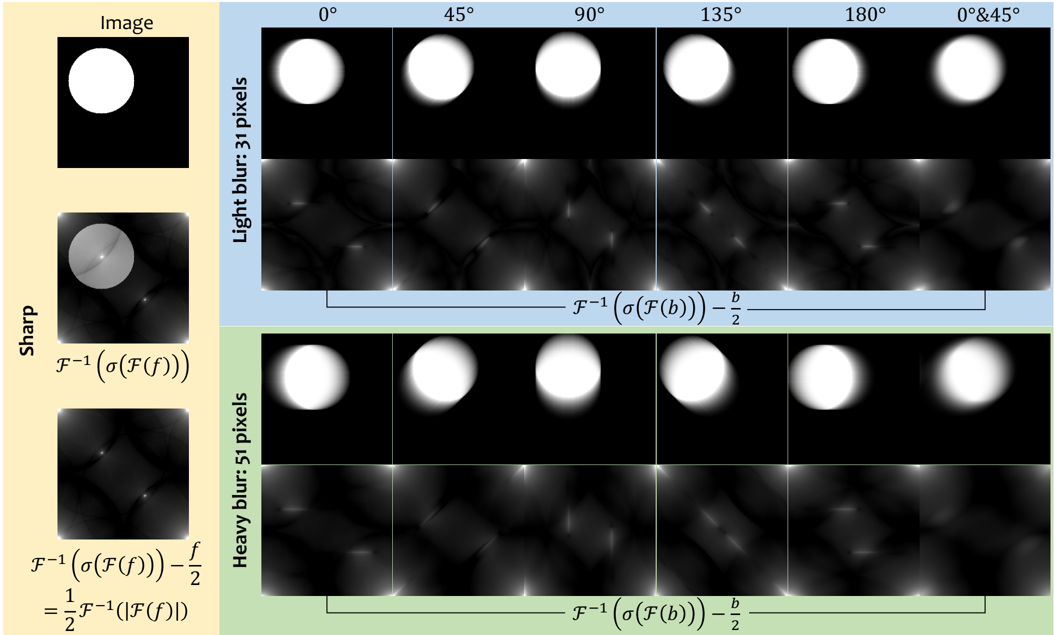

Figure 3: We use a circle image as an example for sharp image f. The image is blurred (b) by linear kernels with motion blur of different directions $\{0°, 45°, 90°, 135°, 180°\}$, different blur levels and a mixture of blur. Blur kernel-like images are calculated by the equations below the image. The middle image of the sharp image is visualized for reference. See more details in the supplementary material about equations $\mathcal{F}^{-1}(\sigma(\mathcal{F}(f))) - f/2$ and $\mathcal{F}^{-1}(\sigma(\mathcal{F}(b))) - b/2$ (best viewed by zoom-in on screen).

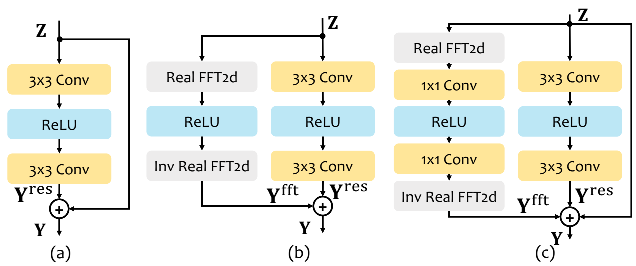

Figure 4: (a) ResBlock. (b) Insert simple FFT-ReLU stream into ResBlock. (c) Proposed Res FFT-ReLU Block, where the bottom stream is termed as FFT-ReLU stream.

$X[N-k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}(N-k)n} = X^*[k]$. The same applies to 2D DFT, which performs sequential row and column 1D DFT on a 2D signal whose size is $M \times N$, *i.e.*, $X[M-u, N-v] = X^*[u,v]$. Since the results of a real array's DFT has symmetric properties, the right half of the results can be derived from the left half. The FFT algorithms reduce the complexity and calculates the DFT in a more efficient way. Let $\mathbf{Z} \in \mathbb{R}^{H \times W \times C}$ be the input feature volume, where H, W, and C indicate the height, width and channel of the feature volume. The left stream in Res FFT-ReLU Block, called FFT-ReLU stream (see Fig. 4(c)) is processed as follows:

(1) computes 2D real FFT of $\mathbf{Z}$ and obtain $\tilde{\mathbf{Z}} = \mathcal{F}(\mathbf{Z}) \in \mathbb{C}^{H \times W/2 \times C}$, where $\mathbb{C}$ means complex domain.

(2) uses two stacks of 1×1 convolution layers (convolution operator $\otimes$) with a ReLU layer in between: $h(\tilde{\mathbf{Z}}; \mathbf{\Theta}^{(1)}, \mathbf{\Theta}^{(2)}) = \text{ReLU}(\tilde{\mathbf{Z}} \otimes \mathbf{\Theta}^{(1)}) \otimes \mathbf{\Theta}^{(2)} \in \mathbb{C}^{H \times W/2 \times C}$, where $\mathbf{\Theta}^{(1)}, \mathbf{\Theta}^{(1)} \in \mathbb{C}^{C \times C}$ are parameters in complex values, and $h(\cdot; \mathbf{\Theta}^{(1)}, \mathbf{\Theta}^{(2)})$ is the network block parameterized by $\mathbf{\Theta}^{(1)}$ and $\mathbf{\Theta}^{(2)}$. We apply ReLU to the real and imaginary parts respectively.

(3) applies inverse 2D real FFT to transform $h(\tilde{\mathbf{Z}}; \mathbf{\Theta}^{(1)}, \mathbf{\Theta}^{(2)})$ back to spatial domain: $\mathbf{Y}^{\text{fft}} = \mathcal{F}^{-1}\left(h(\tilde{\mathbf{Z}}; \mathbf{\Theta}^{(1)}, \mathbf{\Theta}^{(2)})\right) \in \mathbb{R}^{H \times W \times C}$.

Then the final output of Res FFT-ReLU Block is calculated via $\mathbf{Y} = \mathbf{Y}^{\text{fft}} + \mathbf{Y}^{\text{res}} + \mathbf{Z}$, where $\mathbf{Y}^{\text{res}}$ uses the same computation as that in the original ResBlock. The code of realizing $\Theta^{(1)}$ on $\mathbf{Z}$ is:

```
w_real = nn.Parameter(torch.Tensor(C, C))
w_imag = nn.Parameter(torch.Tensor(C, C))
w_c = torch.complex(w_real,w_imag)
Z_o = Z @ w_c # @ is matrix multiplication
```

Global Context Learning Ability

Convolution on Frequency Does Not Bring the Global Context Learning Ability Mathematically, given $x[n]$, which is a sequence of N numbers. For $S \times 1$ convolution on the spatial domain, weights are denoted as $w[a]$, where $a \in \mathcal{N}_n(S)$, and $\mathcal{N}_n(S)$ indicates a $S \times 1$ neighborhood of $x[n]$. We have $y = \sum_a x[n-a] \cdot w[a]$. If we conduct $S \times 1$ convolution on the frequency domain, we have

$y = \mathcal{F}^{-1}(\sum_a X[k-a]w[a]) = \sum_a w[a] \cdot x[n] \cdot e^{j2\pi an/N}$, where $X[k]$ is from Res FFT-ReLU Block section. This indicates that linear convolution operation on frequency domain is only used to extract features, which cannot introduce global context learning ability to the network.

ReLU on Frequency Brings the Global Context Learning Ability for the Network ReLU is a non-linear operation. The output feature on the rth spatial location after converting to the spatial domain is calculated by $x[r] = \frac{1}{N}\sum_{k=0}^{N-1}\left(\sigma(\sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N})e^{j2\pi rk/N}\right)$. Since ReLU is nonlinear, unlike convolution, it is not possible to simplify this equation by eliminating any $x[n]$. Thus, introducing ReLU in the frequency domain brings the global context learning ability for the network. We will show in Table 1 that ReLU in frequency domain not always helps improve performance, whose location matters.

①	②	③	④	⑤	⑥	⑦	⑧	**PSNR**	**Params** (M)
✓	×	×	×	×	×	-	-	28.06	0.30
✓	✓	×	×	×	×	-	-	29.17	0.30
✓	×	✓	×	×	×	-	-	29.77	0.30
✓	×	×	✓	×	×	-	-	29.73	0.30
✓	×	×	×	✓	×	-	-	29.08	0.30
×	×	×	×	✓	×	-	-	29.51	0.30
✓	×	×	×	×	✓	-	-	*27.78*	0.30
-	-	-	-	-	-	×	×	30.30	0.36
-	-	-	-	-	-	✓	×	30.17	0.35
-	-	-	-	-	-	✓	✓	29.96	0.33

Table 1: Ablation on GoPro dataset with RSNet (Nah, Kim, and Lee 2017) for Fig. 5(a) (upper area) and Fig. 5(b) (bottom area). "×" for ⑤ means the left stream is totally removed in Fig. 5(a). Result worse than RSNet is in *italics*.

Experiments

Experimental Setup

Dataset Three datasets are mainly evaluated: GoPro (Nah, Kim, and Lee 2017), HIDE (Shen et al. 2019) and RealBlur (Rim et al. 2020) datasets. Since existing methods adopt different experimental settings, we summarize them and report two groups of results: *(I)* train on 2,103 pairs of blurry and sharp images in GoPro dataset, and test on 1,111 image pairs in GoPro (follow (Cho et al. 2021)), 2,025 image pairs in HIDE (follow (Zamir et al. 2021)), 980 image pairs in RealBlur-R test set, and 980 image pairs in RealBlur-J test set (follow (Zamir et al. 2021)), respectively; *(II)* train on 3,758 image pairs in RealBlur-R, and test on 980 image pairs in RealBlur-R (follow (Zamir et al. 2021)), and train on 3,758 image pairs in RealBlur-J, and test on 980 image pairs in RealBlur-J (follow (Zamir et al. 2021)). Besides, we also show the effectiveness of Res FFT-ReLU Block on REDS dataset (Nah et al. 2021), with 24,000 and 3,000 images for training and testing, respectively (follow (Tu et al. 2022)).

Loss Function We unify the loss function for all experiments. $\hat{\mathbf{S}}$, $\mathbf{S}$ and ε denote the predicted sharp image, the groundtruth sharp image, and a constant value 10^{-3}, respectively. Two kinds of loss functions are adopted: (1) Charbonnier loss (Zamir et al. 2021): $\mathcal{L}_c = \sqrt{||\hat{\mathbf{S}} - \mathbf{S}||^2 + \varepsilon^2}$, and (2) Frequency Reconstruction (FR) loss (Cho et al. 2021; Tu et al. 2022): $\mathcal{L}_{fr} = ||\mathcal{F}(\hat{\mathbf{S}}) - \mathcal{F}(\mathbf{S})||_1$. Finally, the loss function is $\mathcal{L} = \mathcal{L}_c + \alpha_1 \mathcal{L}_{fr}$, where α_1 is a tradeoff-parameter and is empirically set to 0.01.

Implementation Details We adopt the training strategy used in MPRNet (Zamir et al. 2021) unless otherwise specified. *I.e.*, the network training hyperparameters (and the default values we use) are patch size (256× 256), batch size (16), training epoch (3,000), optimizer (Adam (Kingma and Ba 2015)), initial learning rate (2×10^{-4}). The learning rate is steadily decreased to 1×10^{-6} using the cosine annealing strategy (Loshchilov and Hutter 2017). Following (Zamir et al. 2021), horizontal and vertical flips are randomly applied on patches for data augmentation. For testing, we adopt the same dataset slicing crop method as used in SDWNet (Zou et al. 2021), where we utilize a step of 256 to perform 256×256 size sliding window slicing, and compensate slicing on the edge part.

Evaluation metric The average performance of PSNR and SSIM over all testing sets are computed by using the official software released by (Zamir et al. 2021). We report number of parameters, FLOPs, and testing time per image (see supplementary material) on on a workstation with Intel Xeon Gold 6240C CPU, NVIDIA GeForce RTX 3090 GPU.

Position Ablation on Simple-FFT-ReLU Stream

We discuss how the position of simple-FFT-ReLU stream on the ResBlock changes the deblurring performance. Experiments are conducted on GoPro dataset (Group I setting). The backbone architecture we use is a simplified version of DeepDeblur (Nah, Kim, and Lee 2017), which contains 16 ResBlocks, and termed as RSNet in the paper. As shown in Fig. 5(a), the position of simple-FFT-ReLU is changed *w.r.t.* ResBlock. Results are reported in Table 1. If we insert simple-FFT-ReLU in postion ⑥, the performance drops, compared with RSNet (27.78 vs. 28.06). If we replace the *identity* mapping by the simple-FFT-ReLU stream, PSNR increases compared with RSNet (29.51 vs. 28.06). This verifies our claim on the effectiveness of replacing *identity* mapping by simple FFT-ReLU stream, as marked in gray in Table 1. Based on this observation, we further change the threshold for filtering out frequencies. We conduct simple statistics and find that the real/imag values after FFT are mainly around $-10,000(1+j)$ to $+10,000(1+j)$. Thus, instead of setting $0+j0$ as the threshold, we try different thresholds, and also apply inverse ReLU (filter out frequency components larger than $0+j0$). Results are shown in Table 2. Inverse ReLU achieves similar results with ReLU (29.47 vs. 29.51). A positive threshold such as $+100(1+j)$ achieves better performance. Replacing ReLU by selecting high-/low- frequency components are not helpful at all, although all are non-linear operations. We also conduct ablation study on the position of 1×1 convolution with simple-FFT-ReLU (see Fig. 5 (b)). Results are shown in Table 1.

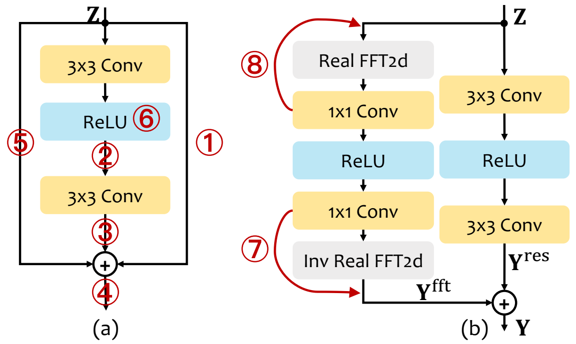

Figure 5: Ablation on (a) the position of simple-FFT-ReLU, and (b) the position of 1×1 convolution with simple-FFT-ReLU, where ① means we keep the *identity* mapping, and ⑥ indicates replacing ReLU with simple-FFT-ReLU. The others means the position where we insert simple-FFT-ReLU. ⑦ and ⑧ mean we put 1×1 Conv before Real FFT2d or after inv Real FFT2d.

Thre	Freq	PSNR
$-1000(1+j)$	-	28.71
$+1000(1+j)$	-	29.32
$-100(1+j)$	-	28.78
$+100(1+j)$	-	29.60
inv. ReLU	-	29.47
-	HF (1/8)	*27.81*
-	LF (1/8)	*27.83*

Table 2: Ablation on GoPro dataset with RSNet (Nah, Kim, and Lee 2017), for selecting frequency by different thresholds (upper), or replacing ReLU by high/low frequency selection (bottom) for the setting highlighted in Table 1.

Ablation Study on FFT-ReLU Stream

Quantitative Results We conduct extensive ablation study on GoPro dataset (Group I setting) to have a thorough investigation on the proposed FFT-ReLU stream. We can summarize the following conclusions from Table 3. (1) Using RSNet with only our FFT-ReLU stream leads to obvious performance drop, compared with RSNet (25.64 vs. 28.06), but adding FFT-ReLU stream to RSNet leads to significant improvement (30.30 vs. 28.06). This indicates that our FFT-ReLU stream is an add on stream, which should be trained in conjunction with the pixel-level spatial domain CNN. We will illustrate the reason by visualizing neurons in the next subsection. (2) It is expected to see that w/o ReLU w/ two 1×1 convolution layers, the performance does not change too much, compared with RSNet (28.59 vs. 28.06). (3) If we change the complex convolution into real convolution, the performance drops a little due to the decrease of parameter numbers (29.91 vs. 30.30). (4) Noted that we propose a complex convolution in frequency domain. An alternative is to first concatenates the real part $\mathcal{R}(\mathcal{F}(\mathbf{Z}))$ and the imaginary part $\mathcal{I}(\mathcal{F}(\mathbf{Z}))$ along the channel dimension to acquire $\tilde{\mathbf{Z}} = \mathcal{R}(\mathcal{F}(\mathbf{Z})) \odot_C \mathcal{I}(\mathcal{F}(\mathbf{Z})) \in \mathbb{R}^{H\times W/2\times 2C}$,

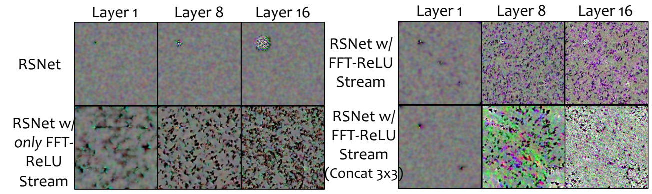

Figure 6: Visualization of example features of three layers of different networks. In each layer, to maximize the neuron on location $[\frac{1}{4}H, \frac{1}{4}W]$, we show visualizations from one random gradient descent run for channel $C = 16$.

ResBlock		FFT-ReLU Stream					PSNR	Params
Z	$\mathbf{Y}^{res}$	FFT	Conv	ReLU	Conv	iFFT	(dB)	(M)
✓	✓	×	×	×	×	×	28.06	0.30
×	×	✓	✓	✓	✓	✓	*25.64*	0.07
✓	×	✓	✓	✓	✓	✓	*26.00*	0.07
✓	✓	×	✓	✓	✓	×	28.16	0.33
✓	✓	✓	×	✓	×	✓	29.08	0.30
✓	✓	✓	✓	×	✓	✓	28.59	0.36
✓	✓	✓	✓	×	×	✓	28.60	0.33
✓	✓	✓	Real	✓	Real	✓	29.91	0.33
✓	✓	✓	✓	✓	×	✓	30.06	0.33
✓	✓	✓	✓	✓	✓	✓	30.30	0.36
✓	✓	✓	✓	GeLU	✓	✓	30.42	0.36
✓	✓	✓	Concat	✓	Concat	✓	30.32	0.43
✓	✓	✓	3×3	✓	3×3	✓	31.15	1.48

Table 3: Ablation on FFT-ReLU stream on GoPro dataset with RSNet (Nah, Kim, and Lee 2017). "Real" means the weights in 1×1 convolution are all real values, instead of the default complex values. "Concat" means after applying FFT, we concatenate the real part $\mathcal{R}(\mathcal{F}(\mathbf{Z}))$ and the imaginary part $\mathcal{I}(\mathcal{F}(\mathbf{Z}))$ along the channel dimension, so the weights of subsequent convolution operations are all real values. Gray areas indicate RSNet and RSNet w/ FFT-ReLU stream. Results worse than RSNet are in *italics*.

where $\odot_C$ represents concatenation through the channel dimension. Then we can simply use two stacks of 1×1 convolution layers (convolution operator $\otimes$) with a ReLU layer in between: $h(\tilde{\mathbf{Z}}; \mathbf{\Theta}^{(1)}_{\text{real}}, \mathbf{\Theta}^{(2)}_{\text{real}}) = \text{ReLU}(\tilde{\mathbf{Z}} * \mathbf{\Theta}^{(1)}_{\text{real}}) \otimes \mathbf{\Theta}^{(2)}_{\text{real}} \in \mathbb{R}^{H\times W/2\times 2C}$. The inverse 2D real FFT can be applied afterwards. We term this alternative as "Concat". Using Concat, the weights in convolution are all real values, and the performance does not change compared with our complex convolution (30.32 vs. 30.30), but the number of parameters increases. (5) We further replace 1×1 convolution by 3×3 convolution, which can be easily implemented in "Concat" setting. A further improvement is oberved (31.15 vs. 30.32). But, Concat 3×3 increases the model parameters by 4 times, compared with Concat 1×1.

Visualizations of Neurons We visualize neurons to better understand how our FFT-ReLU stream works. In (Yosinski et al. 2015), the learned features computed by individual neurons at any layer of the network can be visualized by generating the input image such that the corresponding neurons activation value is maximized. Similarly, given a feature vol-

Model	FFT	GoPro	HIDE	Params	FLOPs
DeepDeblur	×	31.15	29.17	11.71	336.03
	✓	32.37	30.86	12.65	363.19
RSNet	×	28.06	26.01	0.30	19.48
	✓	30.30	28.87	0.36	23.84
U-Net	×	29.20	27.15	0.62	12.18
	✓	30.39	28.71	0.76	14.86
MPRNet -small	×	31.09	29.66	2.14	80.70
	✓	32.50	30.82	2.43	94.24
MIMO-UNet	×	31.90	29.62	6.80	67.17
	✓	32.71	30.85	8.17	80.21
NAFNet32	×	32.95	30.60	17.1	16.00
	✓	33.12	30.76	17.8	23.10

Table 4: Evaluation of Res FFT-ReLU Block (Group I setting) in PSNR. FFT means the ResBlock is replaced by Res FFT-ReLU Block. × means the original architecture. All models are trained by ourselves for fair comparison.

ume $\mathbf{Z}_i \in \mathbb{R}^{H\times W\times C}$ in Layer i, where H, W and C mean the height, width and the channel, we show visualizations of neurons on spatial location of $[\frac{1}{4}H, \frac{1}{4}W]$ from one random gradient descent run for $C = 16$ in Layer 1, 8, 16, using a public repository[1]. As shown in Fig. 6, to maximize the neuron on $[\frac{1}{4}H, \frac{1}{4}W]$, RSNet only gathers information from its local neighborhood. RSNet w/ *only* FFT-ReLU stream can learn global context, but using only FFT-ReLU stream lacks pixel-level localization ability, *i.e.*, there is not a clear activation region on $[\frac{1}{4}H, \frac{1}{4}W]$. This explains the reason why it can only obtain 26.00 dB in PSNR in Table 3. RSNet w/ FFT-ReLU stream can learn both kernel-level and pixel-level representations. Another interesting observation is that though FFT-ReLU stream has the ability to learn global context, in the lower layers, *e.g.*, Layer 1, local information is more important, like Transformer (Raghu et al. 2021). Other details are provided in the supplementary material.

Evaluation of Res FFT-ReLU Block

Quantitative Results Res FFT-ReLU Block is plug-and-play. We plug it into various architectures: DeepDeblur (Nah, Kim, and Lee 2017); U-Net (one backbone network used in MPRNet (Zamir et al. 2021)); MPRNet-small (Zamir et al. 2021), whose number of channels is three times smaller than original MPRNet due to limited computation resource; MIMO-UNet (Cho et al. 2021); NAFNet (Chen et al. 2022). NAFNet uses a different ResBlock, and we show the building block of NAFNet w/ Res FFT-ReLU Block in the supplementary material. PSNRs on GoPro and HIDE datasets in Group I setting are summarized in Table 4. Replacing ResBlock by our Res FFT-ReLU Block leads to remarkable performance gains in various architectures. Besides, RSNet and RSNet w/ Res FFT-ReLU Block on **REDS** dataset are tested. Results are 26.78 and 27.79↑**1.01** dB respectively.

	GoPro		HIDE		RealBlur-R/J	
Method	PSNR	SSIM	PSNR	SSIM	PSNR	PSNR
CVPR						
DeepDeblur(17)	29.08	0.914	25.73	0.874	32.51	27.87
SRN(18)	30.26	0.934	28.36	0.915	35.66	28.56
DMPHN(19)	31.20	0.940	29.09	0.924	35.70	28.42
DBGAN(20)	31.10	0.942	28.94	0.915	33.78	24.93
MPRNet(21)	32.66	0.959	30.96	0.939	35.99	28.70
Restormer(22)	32.92	0.961	31.22	0.942	36.19	28.96
Uformer(22)	33.06	0.967	30.90	0.953	36.19	29.09
ICCV						
DeblurGANv2(19)	29.55	0.934	26.61	0.875	35.26	28.70
MIMO-UNet+(21)	32.45	0.957	29.99	0.930	35.54	27.63
ECCV						
MT-RNN(20)	31.15	0.945	29.15	0.918	35.79	28.44
NAFNet64(22)	33.69	0.967	31.32	0.943	-	
Ours						
FMIMO-UNet	33.08	0.962	31.19	0.943	35.96	28.72
FMIMO-UNet+	33.52	0.965	**31.66**	0.946	36.11	28.88
FNAFNet32	33.12	0.962	30.76	0.938	36.07	28.78
FNAFNet64	**33.85**	0.967	31.12	0.944	-	
¶MPRNet	-		-		39.31	31.76
¶FMIMO-UNet	-		-		**40.01**	**32.65**

Table 5: Comparison on GoPro, HIDE and RealBlur datasets (Group I and II settings). Group II results are with ¶. Numbers in the bracket indicate the published year.

Evaluation of FMIMO-UNet and FNAFNet

MIMO-UNet and NAFNet are most recent ResBlock-based models with remarkable speed advantages among existing networks. We design FMIMO-UNet based on MIMO-UNet and MIMO-UNet+, acquiring FMIMO-UNet and FMIMO-UNet+. We design FNAFNet based on NAFNet. FMIMO-UNet and FNAFNet are compared with other state-of-the-arts in Table 5. Res FFT-ReLU block is only added to a few encoder/decoder blocks. See detailed analysis, computational cost comparisons and visualizations in the supplementary material.

Conclusion

In this paper, we reveal an intriguing phenomenon that frequency selection can provide faithful information about the blur pattern. With this viewpoint, we propose a plug-and-play block called Res FFT-ReLU Block based on FFT-ReLU stream. Res FFT-ReLU Block allows the image-wide receptive field which is able to capture the long-term interaction. Plugging Res FFT-ReLU Block into MIMO-UNet and NAFNet achieves remarkable superior performance compared with state-of-the-arts, on three well-known public image deblurring datasets.

[1] https://github.com/utkuozbulak/pytorch-cnn-visualizations

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62101191, 61975056), Shanghai Natural Science Foundation (Grant No. 21ZR1420800), and the Science and Technology Commission of Shanghai Municipality (Grant No. 20440713100, 22DZ2229004).

References

Banham, M. R.; and Katsaggelos, A. K. 1997. Digital image restoration. *IEEE Signal Process. Mag.*, 14(2): 24–41.

Brigham, E. O.; and Morrow, R. E. 1967. The fast Fourier transform. *IEEE Spectrum*, 4(12): 63–70.

Chakrabarti, A.; Zickler, T. E.; and Freeman, W. T. 2010. Analyzing spatially-varying blur. In *Proc. CVPR*.

Chen, L.; Chu, X.; Zhang, X.; and Sun, J. 2022. Simple Baselines for Image Restoration. In *Proc. ECCV*.

Chen, L.; Lu, X.; Zhang, J.; Chu, X.; and Chen, C. 2021. HINet: Half Instance Normalization Network for Image Restoration. In *Proc. CVPR Workshop*.

Chen, X.; Xia, J. J.; Shen, D.; Lian, C.; Wang, L.; Deng, H. H.; Fung, S. H.; Nie, D.; Thung, K.; Yap, P.; and Gateno, J. 2020. One-Shot Generative Adversarial Learning for MRI Segmentation of Craniomaxillofacial Bony Structures. *IEEE Trans. Medical Imaging*, 39(3): 787–796.

Cheng, S.; Wang, Y.; Huang, H.; Liu, D.; Fan, H.; and Liu, S. 2021. NBNet: Noise Basis Learning for Image Denoising With Subspace Projection. In *Proc. CVPR*.

Chi, L.; Jiang, B.; and Mu, Y. 2020. Fast Fourier Convolution. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Proc. NeurIPS*.

Chi, L.; Tian, G.; Mu, Y.; Xie, L.; and Tian, Q. 2019. Fast Non-Local Neural Networks with Spectral Residual Learning. In *Proc. ACM MM*.

Cho, S.; Ji, S.; Hong, J.; Jung, S.; and Ko, S. 2021. Rethinking Coarse-to-Fine Approach in Single Image Deblurring. In *Proc. ICCV*.

Dong, C.; Loy, C. C.; He, K.; and Tang, X. 2016. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2): 295–307.

Gao, H.; Tao, X.; Shen, X.; and Jia, J. 2019. Dynamic Scene Deblurring With Parameter Selective Sharing and Nested Skip Connections. In *Proc. CVPR*.

Guo, Y.; Chen, J.; Wang, J.; Chen, Q.; Cao, J.; Deng, Z.; Xu, Y.; and Tan, M. 2020. Closed-Loop Matters: Dual Regression Networks for Single Image Super-Resolution. In *Proc. CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proc. CVPR*.

Hu, Z.; Cho, S.; Wang, J.; and Yang, M. 2014. Deblurring Low-Light Images with Light Streaks. In *Proc. CVPR*.

Jiang, K.; Wang, Z.; Yi, P.; Chen, C.; Huang, B.; Luo, Y.; Ma, J.; and Jiang, J. 2020. Multi-Scale Progressive Fusion Network for Single Image Deraining. In *Proc. CVPR*.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *Proc. ICLR*.

Kupyn, O.; Budzan, V.; Mykhailych, M.; Mishkin, D.; and Matas, J. 2018. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. In *Proc. CVPR*.

Kupyn, O.; Martyniuk, T.; Wu, J.; and Wang, Z. 2019. DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better. In *Proc. ICCV*.

Liang, J.; Cao, J.; Sun, G.; Zhang, K.; Gool, L. V.; and Timofte, R. 2021. SwinIR: Image Restoration Using Swin Transformer. In *Proc. ICCV Workshop*.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic Gradient Descent with Warm Restarts. In *Proc. ICLR*.

Mei, Y.; Fan, Y.; and Zhou, Y. 2021. Image Super-Resolution With Non-Local Sparse Attention. In *Proc. CVPR*.

Nah, S.; Kim, T. H.; and Lee, K. M. 2017. Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring. In *Proc. CVPR*.

Nah, S.; Son, S.; Lee, S.; Timofte, R.; and Lee, K. M. 2021. NTIRE 2021 Challenge on Image Deblurring. In *Proc. CVPR Workshop*.

Pan, J.; Sun, D.; Pfister, H.; and Yang, M. 2016. Blind Image Deblurring Using Dark Channel Prior. In *Proc. CVPR*.

Pan, L.; Hartley, R. I.; Liu, M.; and Dai, Y. 2019. Phase-Only Image Based Kernel Estimation for Single Image Blind Deblurring. In *Proc. CVPR*.

Park, D.; Kang, D. U.; Kim, J.; and Chun, S. Y. 2020. Multi-Temporal Recurrent Neural Networks for Progressive Non-uniform Single Image Deblurring with Incremental Temporal Training. In *Proc. ECCV*.

Purohit, K.; and Rajagopalan, A. N. 2020. Region-Adaptive Dense Network for Efficient Motion Deblurring. In *Proc. AAAI*.

Purohit, K.; Suin, M.; Rajagopalan, A. N.; and Boddeti, V. N. 2021. Spatially-Adaptive Image Restoration using Distortion-Guided Networks. In *Proc. ICCV*.

Raghu, M.; Unterthiner, T.; Kornblith, S.; Zhang, C.; and Dosovitskiy, A. 2021. Do Vision Transformers See Like Convolutional Neural Networks? In *Proc. NeurIPS*.

Rao, Y.; Zhao, W.; Zhu, Z.; Lu, J.; and Zhou, J. 2021. Global Filter Networks for Image Classification. In *Proc. NeurIPS*.

Rim, J.; Lee, H.; Won, J.; and Cho, S. 2020. Real-World Blur Dataset for Learning and Benchmarking Deblurring Algorithms. In Vedaldi, A.; Bischof, H.; Brox, T.; and Frahm, J., eds., *Proc. ECCV*.

Rippel, O.; Snoek, J.; and Adams, R. P. 2015. Spectral Representations for Convolutional Neural Networks. In *Proc. NeurIPS*.

Shen, Z.; Wang, W.; Lu, X.; Shen, J.; Ling, H.; Xu, T.; and Shao, L. 2019. Human-Aware Motion Deblurring. In *Proc. ICCV*.

Suin, M.; Purohit, K.; and Rajagopalan, A. N. 2020. Spatially-Attentive Patch-Hierarchical Network for Adaptive Motion Deblurring. In *Proc. CVPR*.

Sun, J.; Cao, W.; Xu, Z.; and Ponce, J. 2015. Learning a convolutional neural network for non-uniform motion blur removal. In *Proc. CVPR*.

Suvorov, R.; Logacheva, E.; Mashikhin, A.; Remizova, A.; Ashukha, A.; Silvestrov, A.; Kong, N.; Goka, H.; Park, K.; and Lempitsky, V. 2022. Resolution-robust Large Mask Inpainting with Fourier Convolutions. In *Proc. WACV*.

Tao, X.; Gao, H.; Shen, X.; Wang, J.; and Jia, J. 2018. Scale-Recurrent Network for Deep Image Deblurring. In *Proc. CVPR*.

Tu, Z.; Talebi, H.; Zhang, H.; Yang, F.; Milanfar, P.; Bovik, A.; and Li, Y. 2022. MAXIM: Multi-Axis MLP for Image Processing. In *Proc. CVPR*.

Wang, Z.; Cun, X.; Bao, J.; and Liu, J. 2022. Uformer: A General U-Shaped Transformer for Image Restoration. In *Proc. CVPR*.

Xu, L.; Zheng, S.; and Jia, J. 2013. Unnatural L0 Sparse Representation for Natural Image Deblurring. In *Proc. CVPR*.

Yang, Y.; and Soatto, S. 2020. FDA: Fourier Domain Adaptation for Semantic Segmentation. In *Proc. CVPR*.

Yosinski, J.; Clune, J.; Nguyen, A. M.; Fuchs, T. J.; and Lipson, H. 2015. Understanding Neural Networks Through Deep Visualization. *CoRR*, abs/1506.06579.

Yuan, Y.; Su, W.; and Ma, D. 2020. Efficient Dynamic Scene Deblurring Using Spatially Variant Deconvolution Network With Optical Flow Guided Training. In *Proc. CVPR*.

Zamir, S. W.; Arora, A.; Khan, S. H.; Hayat, M.; Khan, F. S.; and Yang, M. 2022. Restormer: Efficient Transformer for High-Resolution Image Restoration. In *Proc. CVPR*.

Zamir, S. W.; Arora, A.; Khan, S. H.; Hayat, M.; Khan, F. S.; Yang, M.; and Shao, L. 2020. Learning Enriched Features for Real Image Restoration and Enhancement. In *Proc. ECCV*.

Zamir, S. W.; Arora, A.; Khan, S. H.; Hayat, M.; Khan, F. S.; Yang, M.; and Shao, L. 2021. Multi-Stage Progressive Image Restoration. In *Proc. CVPR*.

Zhang, H.; Dai, Y.; Li, H.; and Koniusz, P. 2019. Deep Stacked Hierarchical Multi-Patch Network for Image Deblurring. In *Proc. CVPR*.

Zhang, J.; Pan, J.; Ren, J. S. J.; Song, Y.; Bao, L.; Lau, R. W. H.; and Yang, M. 2018a. Dynamic Scene Deblurring Using Spatially Variant Recurrent Neural Networks. In *Proc. CVPR*.

Zhang, K.; Luo, W.; Zhong, Y.; Ma, L.; Stenger, B.; Liu, W.; and Li, H. 2020. Deblurring by Realistic Blurring. In *Proc. CVPR*.

Zhang, Y.; Tian, Y.; Kong, Y.; Zhong, B.; and Fu, Y. 2018b. Residual Dense Network for Image Super-Resolution. In *Proc. CVPR*.

Zhong, Z.; Shen, T.; Yang, Y.; Lin, Z.; and Zhang, C. 2018. Joint Sub-bands Learning with Clique Structures for Wavelet Domain Super-Resolution. In *Proc. NeurIPS*.

Zou, W.; Jiang, M.; Zhang, Y.; Chen, L.; Lu, Z.; and Wu, Y. 2021. SDWNet: A Straight Dilated Network with Wavelet Transformation for Image Deblurring. In *Proc. ICCV Workshop*.