

Deep Learning Image Deblurring Tutorial

Keng Seng Ho

June 2025

Contents

1	Introduction [21]	5
1.1	Blur model	5
1.2	Blur categories	5
1.2.1	Motion blur	5
1.2.2	Out-of-Focus Blur	5
1.2.3	Gaussian Blur	5
1.2.4	Mixed Blur	5
1.3	Traditional Approaches	6
1.4	Deep Learning-Based Approaches	6
2	Early Deep Learning-based Methods	7
2.1	Kernel-based	7
2.1.1	Deep Convolutional Neural Network for Image Deconvolution [18]	7
2.1.2	Learning to Deblur [13]	9
2.2	End-to-End Architectures	13
2.2.1	Deep multi-scale convolutional neural network for dynamic scene deblurring [12]	13
2.2.2	Scale-recurrent Network for Deep Image Deblurring [15]	17
2.2.3	DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks [8]	20
2.2.4	DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better [9]	25
3	UNet-based End-to-End Architectures	29
3.1	CNN-Single stage	29
3.1.1	Rethinking Coarse-to-Fine Approach in Single Image Deblurring [4]	29
3.1.2	Simple Baselines for Image Restoration [1]	33
3.1.3	Intriguing Findings of Frequency Selection for Image Deblurring [11]	37
3.2	CNN-Multi stage	41
3.2.1	Multi-Stage Progressive Image Restoration [20]	41
3.2.2	HINet: Half Instance Normalization Network for Image Restoration [2]	45
3.3	Self-attention	48
3.3.1	Restormer: Efficient Transformer for High-Resolution Image Restoration [19]	48
3.3.2	Efficient Frequency Domain-based Transformers for High-Quality Image Deblurring [6]	51
3.4	MLP	55
3.4.1	MAXIM:Multi-Axis MLP for Image Processing [16]	55
3.5	SSM	60
3.5.1	Efficient Visual State Space Model for Image Deblurring [7]	60
4	UNet-based with Degradation representation learning	63
4.1	CNN-based	63
4.1.1	Frequency-domain Learning with Kernel Prior for Blind Image Deblurring [14]	63
4.2	GAN-based	66
4.2.1	Learning Degradation Representations for Image Deblurring [10]	66
4.3	Flow-based	72
4.3.1	Self-supervised Non-uniform Kernel Estimation with Flow-based Motion Prior for Blind Image Deblurring [5]	72
4.4	Diffusion-based	76
4.4.1	Hierarchical Integration Diffusion Model for Realistic Image Deblurring [3]	76
5	Dataset [17]	80
6	Metrics [17]	81
6.1	Mean square error (MSE)	81
6.2	Peak signal noise ratio (PSNR)	81
6.3	Structural similarity index measure (SSIM)	81

Abstract

Image deblurring is a long-standing low-level computer vision problem, aiming to recover a sharp image from a blurred observation. With the widespread availability of imaging devices, captured images are often affected by various sources of blur, such as camera shake, object motion, and limited depth of field. To address this, a significant amount of research has focused on post-processing techniques to restore high-quality, sharp images. While traditional model-based approaches have achieved promising results, they often struggle to generalize to complex real-world scenarios.

In recent years, the rapid development of deep learning has brought substantial advancements to image deblurring. This tutorial provides an overview of recent deep learning-based methods for image deblurring, including representative papers, benchmark datasets, and performance metrics. We aim to offer a systematic summary to facilitate a better understanding of current progress in the field.

1 Introduction [21]

Image deblurring is a classical low-level computer vision task that aims to recover a sharp image from a blurry one, thereby enhancing the visual quality of the image. Deblurring also plays an essential role in various high-level vision applications such as object detection, autonomous driving, and biomedical image analysis. To study this problem, it is necessary to formulate a model that describes how a blurry image is generated.

1.1 Blur model

The degradation process of a blurry image is commonly modeled as (eq.1):

$$B = \Phi(K, I) + N, \quad (1)$$

where $\Phi(\cdot)$ is the blur function (convolution is always used), B is blurry image, K is blur kernel (Point Spread Function), I is sharp image, and N is noise (Additive White Gaussian Noise).

The blur kernel K can be either space-invariant or space-variant. If K is known, the task is referred to as non-blind deblurring; otherwise, it is known as blind deblurring.

1.2 Blur categories

Blurring can arise from various physical and optical causes. It is typically categorized into four major types (Motion blur, Out-of-Focus blur, Gaussian blur and Mixed blur).

1.2.1 Motion blur

During image acquisition, photons are integrated over a finite exposure time. When the exposure time is long relative to object or camera motion, motion blur occurs.

- Space-invariant motion blur: Typically caused by uniform camera shake.
- Space-variant motion blur: Results from non-uniform motion, such as depth-dependent object movement or rotational motion with varying speed.

1.2.2 Out-of-Focus Blur

Caused by the distance between the object or scene and the camera's focal plane. The amount of blur depends on whether the object is at the true focus or within the depth of field. This type of blur is generally space-variant, as the PSF $K(x, y)$ varies across spatial locations.

1.2.3 Gaussian Blur

This type of blur results from the convolution of the image with a Gaussian Kernel (eq.2):

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (2)$$

where x and y are the distance from the origin in the horizontal and vertical axis, respectively, σ is the standard deviation.

1.2.4 Mixed Blur

A combination of the above types of blur. In real-world scenarios, blur is often caused by a mixture of multiple factors such as camera shake, object motion, and defocus, making the deblurring task highly challenging.

1.3 Traditional Approaches

Image deblurring is inherently an ill-posed inverse problem. Multiple plausible sharp images can produce the same blurry observation, increasing the difficulty of recovering the true solution. Conventional (non-deep learning) methods often adopt filter-based frameworks combined with Maximum a Posteriori (MAP) estimation, where carefully designed priors are introduced to regularize the solution space.

Although traditional approaches can perform well in some cases, they tend to fail under severe or complex real-world blur conditions. One key limitation is the reliance on handcrafted priors, which are often difficult to formulate accurately and may lack generalizability across diverse scenarios.

1.4 Deep Learning-Based Approaches

Due to the limitations of traditional methods and the rapid progress of deep learning, learning-based techniques have become dominant in recent years. Deep models excel at feature learning and nonlinear mapping, allowing them to capture complex blur patterns and representations from data. These data-driven approaches have demonstrated superior performance and robustness in challenging deblurring tasks.

Early deep learning methods often relied on kernel-based frameworks, where networks were trained to approximate deconvolution or estimate blur kernels in a multi-stage pipeline. However, later studies argued that explicitly estimating the blur kernel may not generalize well to real-world conditions due to kernel estimation inaccuracies, complex or spatially varying blur patterns.

This motivated a shift toward end-to-end learning, where the network directly maps blurry images to their sharp images, bypassing the need for explicit kernel estimation. Among these, the U-Net architecture has emerged as the most widely adopted backbone, and many recent works have enhanced it with components such as CNNs, self-attention, MLPs, and state space models (SSMs). Some works further incorporate degradation representation priors, allowing the model to exploit blur characteristics for more effective restoration.

In the following sections, we provide a comprehensive overview of learning-based image deblurring methods, including early kernel-based and end-to-end architectures, U-Net-based end-to-end designs, and those enhanced with degradation priors. We also summarize commonly used benchmark datasets and evaluation metrics in the image deblurring literature.

2 Early Deep Learning-based Methods

Early deep learning-based approaches for image deblurring can be broadly categorized into kernel-based and end-to-end methods. Kernel-based methods typically decompose the problem into two stages: first estimating the blur kernel, and then performing non-blind deconvolution to recover the sharp image. In contrast, end-to-end methods argue that inaccurate kernel estimation can significantly degrade the restoration performance. Therefore, these approaches bypass kernel estimation entirely and instead train deep models to directly learn a mapping from the blurry image to the sharp image in an end-to-end manner.

2.1 Kernel-based

2.1.1 Deep Convolutional Neural Network for Image Deconvolution [18]

This paper primarily addresses the non-blind deblurring setting, where the blur kernel is assumed to be known. The main idea is to incorporate deep learning techniques into the traditional deconvolution process, leveraging neural networks to mitigate the artifacts that often arise from conventional filtering-based methods.

There are two main contributions of this work:

- The authors propose replacing the traditional deconvolution pipeline with a convolutional neural network, which is decomposed into two sub-networks: a Deconvolution Sub-Network (DCNN) and an Outlier Rejection Sub-Network (ODCNN).
- The study investigates the impact of different parameter initialization strategies within the network. Specifically, it shows that random initialization and initialization based on separable kernel inversion can lead to differences in performance.

The overall network is shown in below (fig.1).

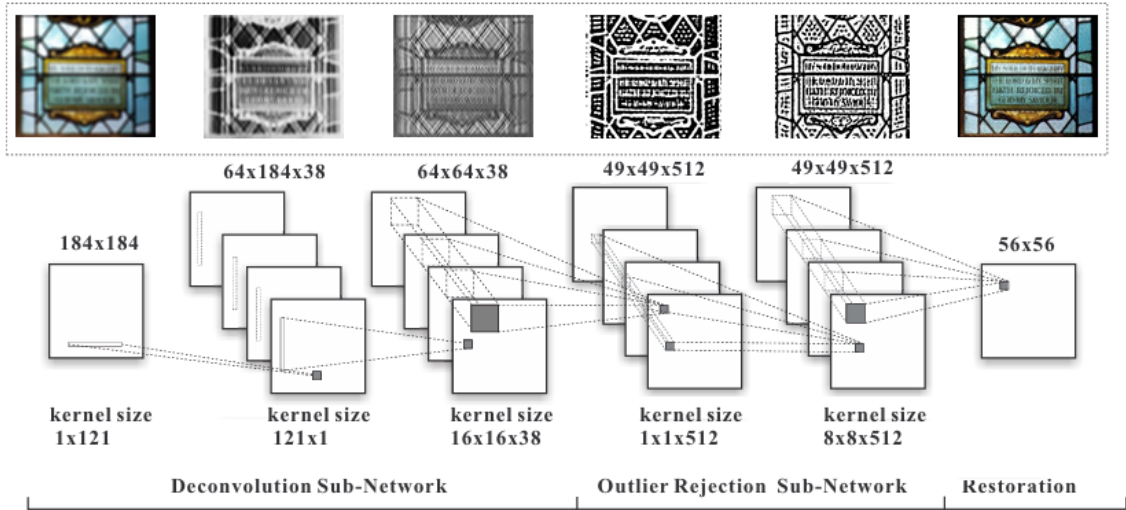


Figure 1: Network architecture for deep deconvolution.

The Deconvolution Sub-Network (DCNN) is primarily motivated by the inverse filter component of the Wiener filter. To begin with, the image formation process under a simple linear blur model can be described as (eq.3):

$$y = x * k, \quad (3)$$

where x is sharp image, k is blur kernel, and y is blurry image. In the frequency domain, this becomes (eq.4):

$$\mathcal{F}(y) = \mathcal{F}(x) \cdot \mathcal{F}(k), \quad (4)$$

where $\mathcal{F}(\cdot)$ is FFT, and $\cdot$ is the element-wise product. The Wiener deconvolution solution can be expressed as (eq.5):

$$x = \mathcal{F}^{-1}\left(\frac{1}{\mathcal{F}(k)}\left\{\frac{|\mathcal{F}(k)|^2}{\mathcal{F}(k)^2 + \frac{1}{\text{SNR}}}\right\}\right) * y = k^\dagger * y, \quad (5)$$

where $\mathcal{F}^{-1}(\cdot)$ is inverse FFT, SNR is the signal-to-noise ratio, $k^\dagger$ denotes the pseudo-inverse kernel. The authors observe that the pseudo-inverse kernel can be decomposed using Singular Value Decomposition (SVD) as $k^\dagger = USV^T$, where U and V are orthogonal matrices and S is a diagonal matrix containing the singular values s_j . Let u_j and v_j denote the j^{th} columns of U and V , respectively. The pseudo deconvolution can then be expressed as (eq.6):

$$k^\dagger * y = \sum_j s_j \cdot u_j * (v_j^T * y), \quad (6)$$

This interpretation reveals that 2D convolution can be approximated as a weighted sum of separable 1D filters. Based on this insight, the deconvolution operation can be learned using a CNN that simulates these filtering operations. The structure of the DCNN is defined as (eq.7):

$$h_3 = W_3 * h_2; \quad h_l = \sigma(W_l * h_{l-1} + b_{l-1}), l \in \{1, 2\}; \quad h_0 = \hat{y}, \quad (7)$$

where W_l is the weight mapping the $(l-1)^{\text{th}}$ layer to the l^{th} one and b_{l-1} is the vector value bias. $\sigma(\cdot)$ is the nonlinear function, which can be sigmoid or hyperbolic tangent. This design corresponds to the Deconvolution Sub-Network shown in (fig.1).

The Outlier Rejection Sub-Network (ODCNN) is inspired by CNN-based denoising architectures. It is specifically designed to suppress undesirable artifacts such as color saturation, noise amplification, and ringing effects—that often emerge after deconvolution.

The overall loss function used to train the network is defined as:

$$\frac{1}{2|N|} \sum_{i \in N} \|f(\hat{y}_i) - \hat{x}_i\|, \quad (8)$$

where $f(\cdot)$ is the network, $|N|$ is the number of image pairs in the sample set, $\hat{y}_i$ is the input blurry image, and $\hat{x}_i$ is the corresponding ground-truth sharp image.

The training process is conducted in a two-stage manner: the DCNN and ODCNN are trained independently, and then their outputs are combined and jointly fine-tuned in a unified pipeline. Notably, the authors find that the initialization strategy of the DCNN has a significant impact on performance—random initialization versus initialization from the separable kernel inversion leads to noticeable differences in training convergence and final restoration quality as shown in the (fig.2).

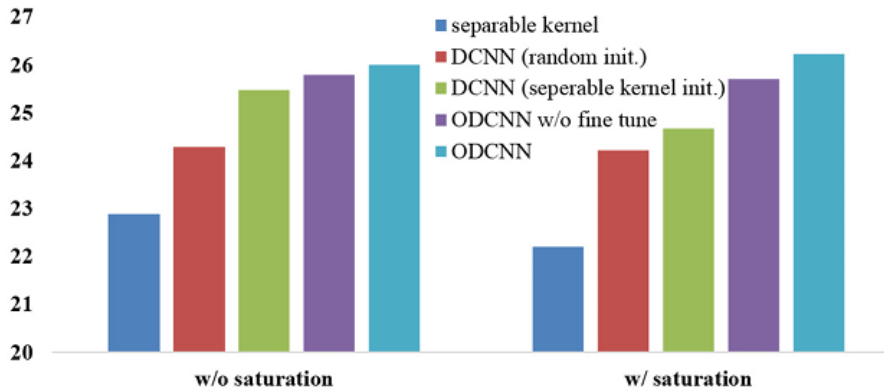


Figure 2: PSNRs produced in different stages of our convolutional neural network architecture.

Quantitative results

kernel type	Krishnan [3]	Levin [7]	Cho [9]	Whyte [10]	Schuler [13]	Schmidt [4]	Ours
disk sat.	24.05dB	24.44dB	25.35dB	24.47dB	23.14dB	24.01dB	26.23dB
disk	25.94dB	24.54dB	23.97dB	22.84dB	24.67dB	24.71dB	26.01dB
motion sat.	24.07dB	23.58dB	25.65 dB	25.54dB	24.92dB	25.33dB	27.76dB
motion	25.07dB	24.47 dB	24.29dB	23.65dB	25.27dB	25.49dB	27.92dB

Figure 3: Quantitative results.

Qualitative results



Figure 4: Qualitative results.

2.1.2 Learning to Deblur [13]

This paper adopts a multi-stage architecture for image deblurring, where each stage consists of three key components: (1) A feature extraction module, (2) A kernel estimation module, and (3) An image estimation module. The overall network is shown in (fig.5)

Among these, the feature extraction module integrates convolutional neural networks (CNNs) to learn informative features from the input image, while the kernel and image estimation modules are formulated using traditional optimization-based techniques.

The main contributions of this paper can be summarized as follows:

- The integration of CNN-based feature extraction into a traditional multi-stage deblurring framework, enabling more expressive representations to guide subsequent kernel and image estimation steps.

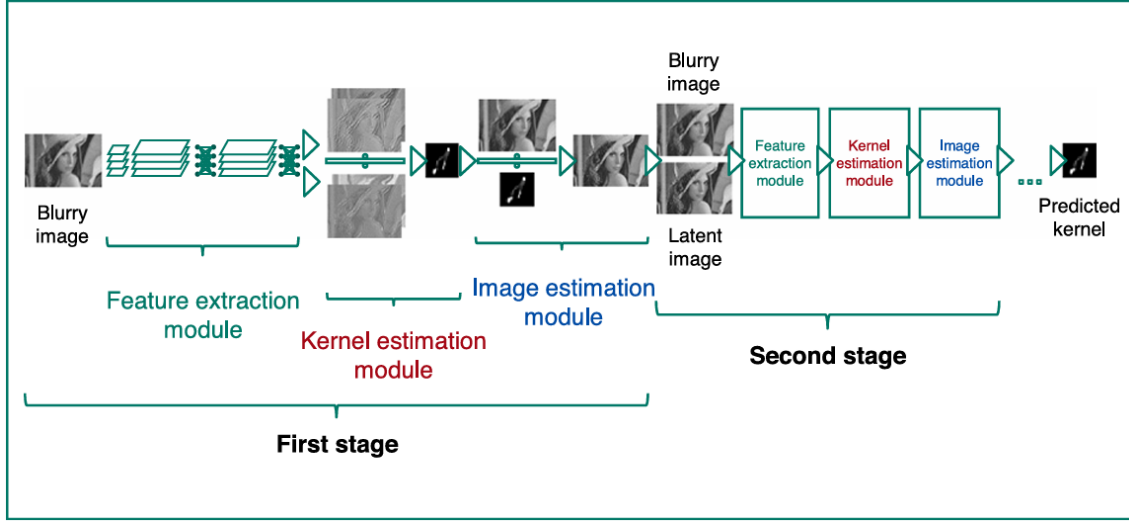


Figure 5: Architecture of proposed blind deblurring network. First the feature extraction module transforms the image to a learned gradient-like representation suitable for kernel estimation. Next, the kernel is estimated by division in Fourier space, then similarly the latent image. The next stages, each consisting of these three operations, operate on both the blurry image and the latent image.

In the feature extraction module, CNNs are employed to extract informative features from the input blurry image. The structure is designed to generate feature maps that resemble gradient-like representations, denoted as $(\tilde{x}_i, \tilde{y}_i)$ and the module shown in (fig.6). These features are computed as (eq.9). This formulation allows the network to capture edge- or gradient-like features in multiple orientations and scales, which are later utilized by the kernel and image estimation modules in subsequent stages.

$$\begin{aligned}\tilde{y}_i &= \sum_j \alpha_{ij} \tanh(f_j * y), \\ \tilde{x}_i &= \sum_j \beta_{ij} \tanh(f_j * y),\end{aligned}\tag{9}$$

where $*$ denotes convolution, $\tanh(\cdot)$ is hyperbolic tangent activation function, f_j is filters of convolution layer, y is blurry image, $\alpha_{i,j}$ and $\beta_{i,j}$ are hidden layers parameters.

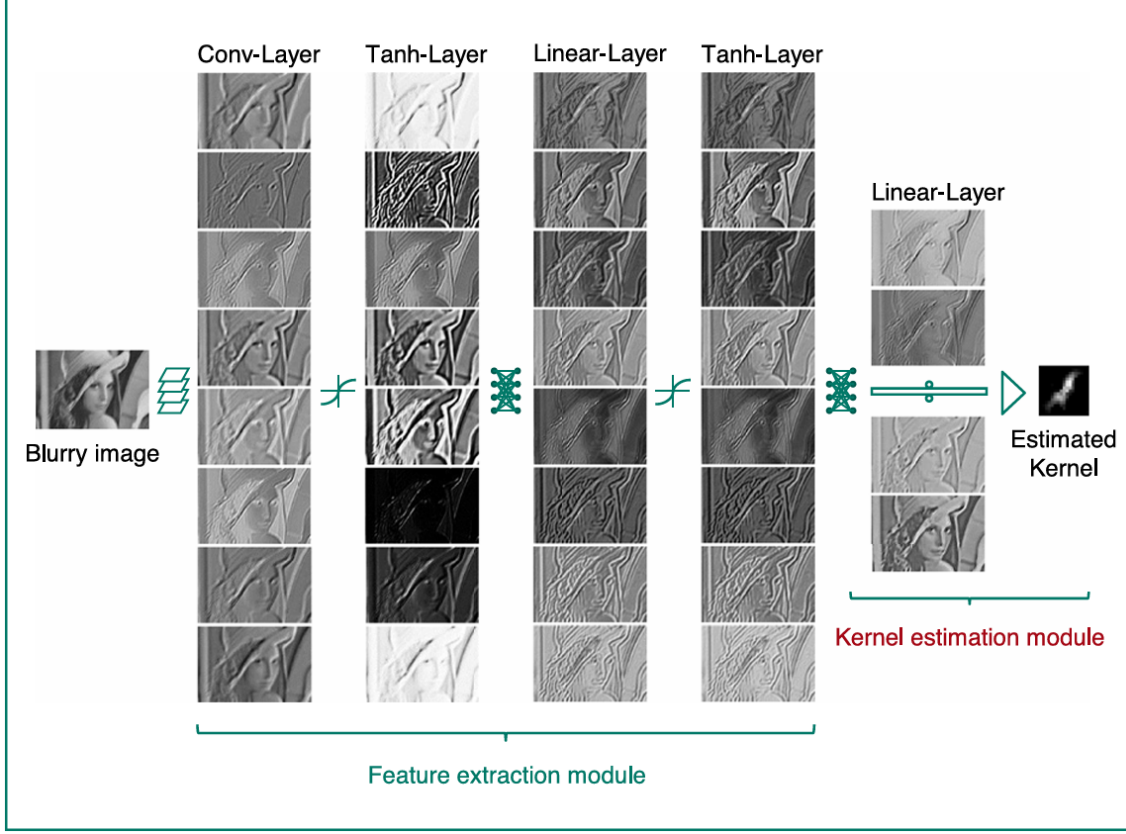


Figure 6: Intermediary outputs of the first stage (see fig.5) of a single-stage NN with the following architecture: eight convolution filters; tanh nonlinearity; linear recombination to eight images; tanh nonlinearity; linear recombination to two sharp and two blurry feature images; kernel estimation.

The kernel estimation module leverages the extracted features $(\tilde{x}, \tilde{y})$ obtained from the feature extraction module to estimate the blur kernel. Specifically, the kernel k is computed by minimizing the following objective function (eq.10):

$$\sum_i \|\tilde{k} * \tilde{x}_i - \tilde{y}_i\|^2 + \beta_k \|\tilde{k}\|^2, \quad (10)$$

This formulation is to find the optimal kernel $\tilde{k}$ that best explains the relationship between the feature pairs $(\tilde{x}, \tilde{y})$, where β_k serves as a regularization term that is learned during training. Assuming noise-free observations and that the kernel has no zeros in its power spectrum, the true sharp image gradients and their blurred counterparts would recover the true kernel when $\beta_k = 0$. The closed-form solution to this minimization is given by (eq.11):

$$\tilde{k} = F^H \frac{\sum_i F \tilde{x}_i \odot F \tilde{y}_i}{\sum_i |F \tilde{x}_i|^2 + \beta_k}, \quad (11)$$

where $F(\cdot)$ is the discrete Fourier transform matrix, $\odot$ is the Hadamard production, and $\bar{v}$ is the complex conjugate of a vector v .

Image Estimation Module is Given the estimated kernel $\tilde{k}$ and the corresponding features, the image estimation module reconstructs the sharp image by solving the following optimization problem:

$$\|\tilde{k} * \tilde{x} - y\|^2 + \beta_x \|\tilde{x}\|^2, \quad (12)$$

where β_x is a trainable regularization parameter. This module ensures that the recovered image is consistent with both the observed blurry image and the estimated blur kernel.

Training strategy that the model is trained in a stage-wise fashion, where each stage is trained sequentially. Specifically, one stage is trained until convergence; then its parameters are frozen

before training the subsequent stage. This progressive training scheme stabilizes optimization and prevents error accumulation across stages. During training, the loss function is defined as the L_2 loss between the estimated kernel and the corresponding ground truth kernel.

Quantitative results

Valley image (Flickr ID)	Xu et al. [20] (dB)	Cont. agnostic training (dB)	Cont. specific training (dB)
fCetj6	24.13	24.61	24.96
hR7YPb	28.59	27.96	28.32
i1GWi8	20.70	20.17	20.91
nz5BxB	22.97	22.28	22.80
nRuiRC	23.32	23.20	23.44
Average	23.94	23.64	24.09

Blackboard image (Flickr ID)	Xu et al. [20] (dB)	Cont. agnostic training (dB)	Cont. specific training (dB)
bkkdz3	26.37	23.63	27.53
btKo6w	20.71	23.87	26.32
dbQBYX	20.88	22.36	24.25
f9jSxK	23.63	21.60	24.41
fK3V16	25.32	23.77	25.22
Average	23.38	23.04	25.55

Figure 7: Quantitative results.

Qualitative results

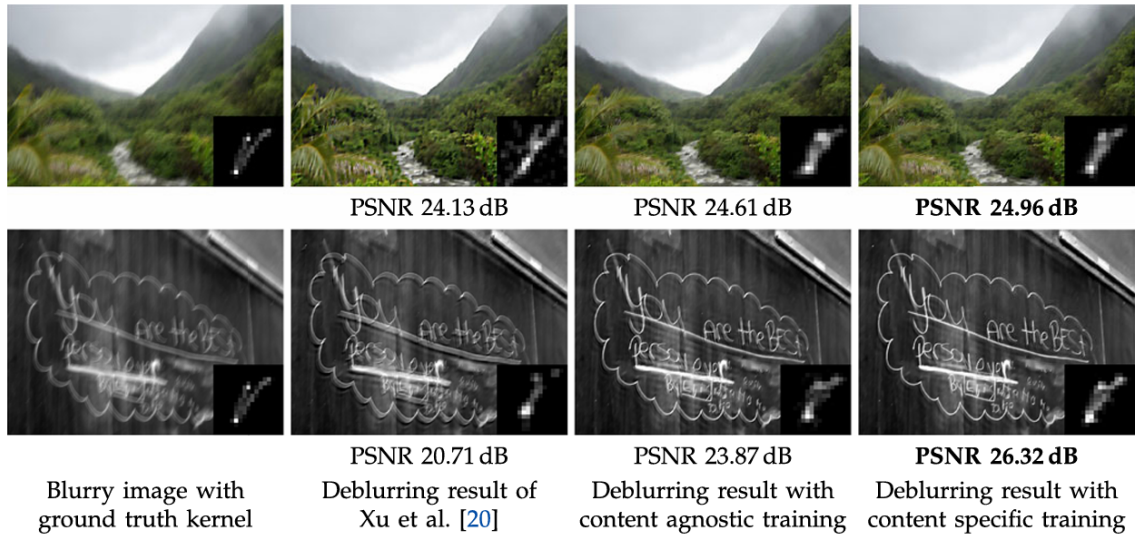


Figure 8: Qualitative results.

2.2 End-to-End Architectures

2.2.1 Deep multi-scale convolutional neural network for dynamic scene deblurring [12]

This paper proposes a coarse-to-fine end-to-end convolutional neural network (CNN) for image deblurring, eliminating the need for stage-wise optimization as adopted in earlier deep learning-based approaches. Instead, the model directly learns to map a blurry image to its corresponding sharp version using ground truth supervision in a fully end-to-end manner.

In addition to the network design, the authors introduce a method for synthesizing realistic motion blur by averaging multiple sharp frames. This technique allows the construction of a more realistic and diverse motion-blurred dataset, which is crucial for supervised training.

The main contributions of this paper are as follow:

- The design of a coarse-to-fine end-to-end CNN architecture that progressively restores image details from low to high resolution.
- The introduction of the GoPro dataset, which provides paired sharp and realistic motion-blurred images for training and evaluation.

The motivation behind adopting a coarse-to-fine architecture is to leverage multi-scale representations of the input image to facilitate more effective deblurring. Specifically, the coarse-scale input having lower resolution and smoother structure is less sensitive to high-frequency noise and artifacts, making it well-suited for providing a stable initial estimate of the sharp image. Once this initial estimate is obtained at the coarse level, the network progressively refines the result at finer scales, where higher-frequency details are incorporated and corrected. This hierarchical refinement enables the network to gradually reconstruct a high-quality sharp image by combining global structure from coarse levels with fine details from higher resolutions.

The overall architecture is illustrated in the following (fig.9):

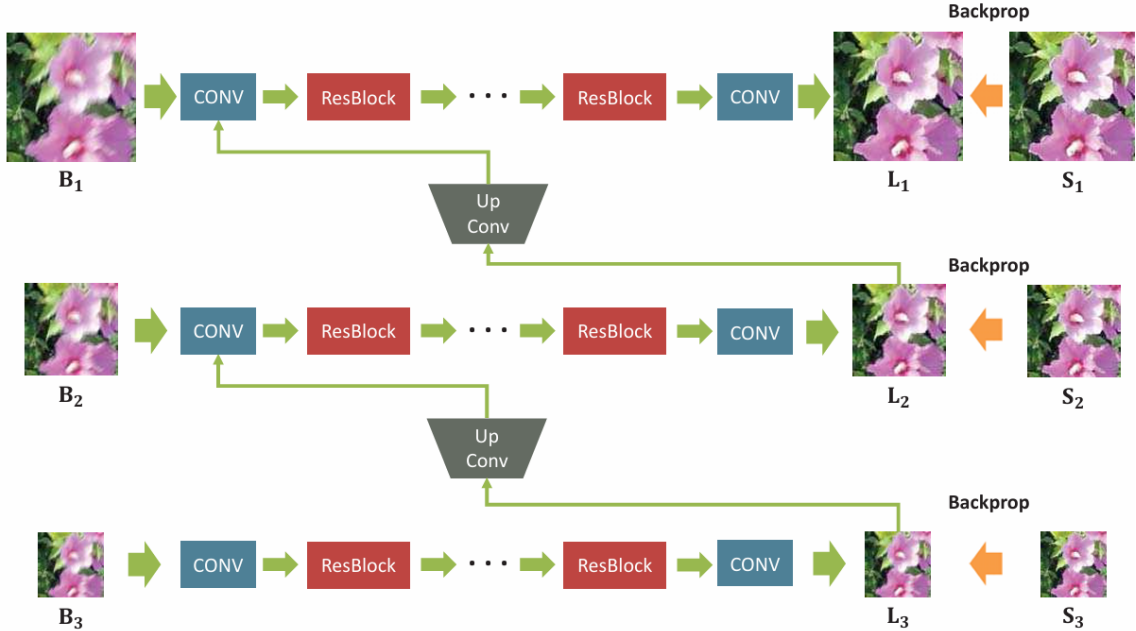


Figure 9: Multi-scale network architecture. B_k , L_k , S_k denote blurry and latent, and ground truth sharp images, respectively. Subscript k denotes k -th scale level in the Gaussian pyramid, which is downsampled to $1/2^k$ scale. Our model takes a blurry image pyramid as the input and outputs an estimated latent image pyramid. Every intermediate scale output is trained to be sharp. At test time, original scale image is chosen as the final result.

The architecture employs a modified version of the residual block (ResBlock) to enhance feature representation during training as shown in (fig.10). Specifically, the authors introduce several adjustments to the standard residual network building block based on empirical observations. Since the model is trained with a mini-batch size of 2, they remove the batch normalization layers, as such small batch sizes may lead to unstable statistics and degrade performance. Additionally, the ReLU activation function is removed from the residual block, as the authors found that omitting it leads to better deblurring results in practice. This simplified yet effective design improves the model’s ability to preserve image structures while reducing undesired artifacts during training and inference.

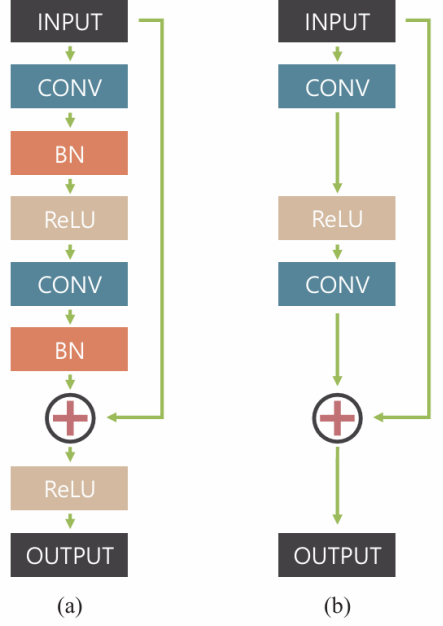


Figure 10: (a) Original residual network building block. (b) Modified building block of our network. We did not use batch normalization layers since we trained model with mini-batch of size 2, which is smaller than usual for batch normalization. We found removing rectified linear unit just before the block output is beneficial in terms of performance empirically.

To enhance the realism of the restored sharp images, the authors incorporate an adversarial network (fig.11) into the training framework. This additional component encourages the deblurring network to produce outputs that are perceptually closer to real sharp images. A discriminator network is introduced to distinguish between the restored image and the corresponding ground truth. Specifically, the discriminator operates on the output from the finest scale of the deblurring network, comparing it against the ground truth sharp image to compute the adversarial loss $\mathcal{L}_{adv}$.

#	Layer	Weight dimension	Stride
1	conv	$32 \times 3 \times 5 \times 5$	2
2	conv	$64 \times 32 \times 5 \times 5$	1
3	conv	$64 \times 64 \times 5 \times 5$	2
4	conv	$128 \times 64 \times 5 \times 5$	1
5	conv	$128 \times 128 \times 5 \times 5$	4
6	conv	$256 \times 128 \times 5 \times 5$	1
7	conv	$256 \times 256 \times 5 \times 5$	4
8	conv	$512 \times 256 \times 5 \times 5$	1
9	conv	$512 \times 512 \times 4 \times 4$	4
10	fc	$512 \times 1 \times 1 \times 1$	-
11	sigmoid	-	-

Figure 11: Model parameters of the discriminator. Every convolution layers are activated with Leaky ReLU layer.

The loss function is composed of two terms: a content loss $\mathcal{L}_{cont}$ and an adversarial loss $\mathcal{L}_{adv}$, which together guide the network to generate both pixel-wise accurate and perceptually realistic sharp images.

Content Loss is given the coarse-to-fine architecture, the content loss is defined as the mean squared error (MSE) between the intermediate outputs of the network at each scale and their corresponding ground truth images. This multi-scale supervision allows the model to learn a coarse-to-fine reconstruction progressively. The content loss is defined as (eq.13):

$$\mathcal{L}_{cont} = \frac{1}{2K} \sum_{k=1}^K \frac{1}{c_k w_k h_k} \|L_k - S_k\|^2, \quad (13)$$

where L_k, S_k denote the model output and ground truth image at scale level k , respectively. The loss at each scale is normalized by the number of channels c_k , width w_k , and the height h_k (i.e. the total number of elements).

Adversarial loss is discriminator taking the output of the finest scale or the ground truth sharp image as input and classifies if it is deblurred image or sharp image, which is defined as (eq.14):

$$\mathcal{L}_{adv} = \mathbb{E}_{S \sim p_{sharp}(S)} [\log D(S)] + \mathbb{E}_{B \sim p_{blurry}(B)} [\log(1 - D(G(B)))], \quad (14)$$

where G and D denote the generator, that is our multi-scale deblurring network in (fig.9) and the discriminator (classifier), respectively. When training, G tries to minimize the adversarial loss while D tries to maximize it.

The overall loss combines both content loss and adversarial loss (eq.15):

$$\mathcal{L}_{total} = \mathcal{L}_{cont} + \lambda \times \mathcal{L}_{adv}, \quad (15)$$

where $\lambda = 1 \times 10^{-4}$ is a weighting factor that balances the contribution of adversarial learning.

Another major contribution of this paper is the introduction of a realistic motion blur synthesis method for generating paired training data. The motivation stems from the nature of motion blur in real-world scenarios. It may be caused either by camera shake or object motion. In the latter case, traditional blur models that rely on spatially-invariant convolution with a single blur kernel are often insufficient. For instance, when only the object is moving while the background remains static, convolution-based blurring would incorrectly blur both foreground and background regions as illustrated in the (fig.12), which does not reflect real-world motion characteristics.

To better approximate the actual image formation process, the authors propose a physically motivated formulation. A blurry image is generated when the camera sensor integrates incoming light over a finite exposure time. This process can be approximated by averaging a sequence of sharp frames captured at a high frame rate during that exposure period. Formally, the blurred image B is modeled as (eq.16):

$$B = g\left(\frac{1}{T} \int_{t=0}^T S(t)dt\right) \simeq g\left(\frac{1}{M} \sum_{i=0}^{M-1} S[i]\right), \quad \text{for } g(x) = x^{1/\gamma}, \quad (16)$$

where T and $S(t)$ denote the exposure time and the sensor signal of a sharp image at time t , respectively. Similarly, M , $S[i]$ are the number of sampled frames and the i^{th} sharp frame signal captured during the exposure time, respectively. g is the CRF that maps a sharp latent signal $S(t)$ into an observed image $\hat{S}(t)$ such that $\hat{S}(t) = g(S(t))$, or $\hat{S}[i] = g(S[i])$. In practice, we only have observed video frames while the original signal and the CRF is unknown. In common practical method is to approximate CRF as a gamma curve with $\gamma = 2.2$

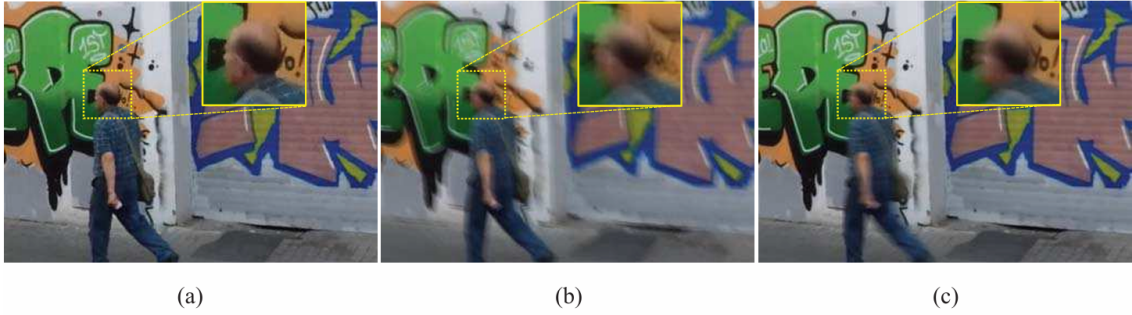


Figure 12: (a) Ground truth sharp image. (b) Blurry image generated by convolving a uniform blur kernel. (c) Blurry image by averaging sharp frames. In this case, blur is mostly caused by person motion, leaving the background as it is. The blur kernel is non-uniform, complex shaped. However, when the blurry image is synthesized by convolution with a uniform kernel, the background also gets blurred as if blur was caused by camera shake. To model dynamic scene blur, our kernel-free method is required.

Quantitative results

Measure	[26]	[15]	Ours		
			$K = 1$	$K = 2$	$K = 3$
PSNR	24.64	23.64	28.93	29.23	29.08
SSIM	0.8429	0.8239	0.9100	0.9162	0.9135
Runtime	20 min	1 hr	7.21 s	4.33 s	3.09 s

Figure 13: Quantitative deblurring performance comparison on the GOPRO dataset. K denotes the scale level.

Measure	[26]	[15]	Ours		
			$K = 1$	$K = 2$	$K = 3$
PSNR	25.22	24.68	25.74	26.02	26.48
MSSIM	0.7735	0.7937	0.8042	0.8116	0.8079

Figure 14: Quantitative comparison on the Kohler dataset.

Qualitative results

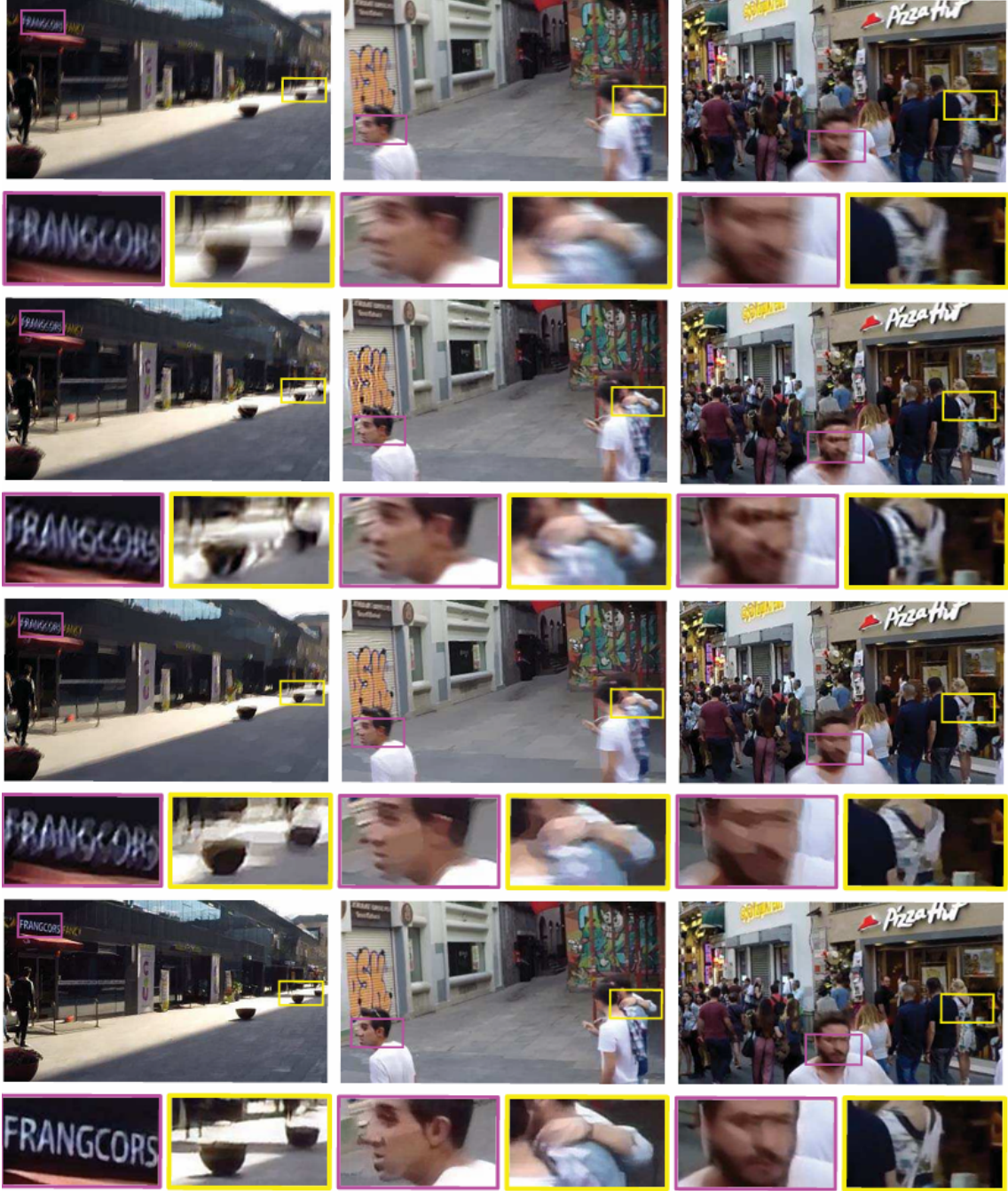


Figure 15: Test results on the GOPRO dataset. From top to bottom: Blurry images, results of Sun et al., results of Kim and Lee, and results of the proposed method.

2.2.2 Scale-recurrent Network for Deep Image Deblurring [15]

This paper builds upon the coarse-to-fine strategy proposed in DeepDeblur, introducing a refined architecture based on an encoder-decoder framework. Specifically, the model consists of three major components: (1) An encoder (EBlock) composed of stacked residual blocks (ResBlocks) to extract deep features and expand the receptive field. (2) A scale-recurrent module using a convolutional LSTM (ConvLSTM) in the latent space to share information across scales. (3) A decoder (DBlock) that progressively reconstructs the sharp image from the encoded representation.

The main contributions of this work are as follows:

- The introduction of a scale-recurrent structure that shares parameters across multiple resolution levels, enabling cross-scale feature propagation and reducing overfitting to specific resolutions or motion patterns.
- An effective exploration of encoder-decoder architectures with ResBlocks for the image deblurring task, demonstrating superior performance with fewer parameters.

Network Architecture Details is using the encoder and decoder components both employ modified residual blocks. Compared to standard ResNet blocks, these ResBlocks are simplified: each contains two convolutional layers, a ReLU activation, and a skip connection, with batch normalization removed for efficiency and stability.

The encoder block (EBlock) consists of one convolutional layer followed by a ReLU and three consecutive ResBlocks. The decoder block (DBlock) includes three ResBlocks, followed by a deconvolution (transposed convolution) layer and a ReLU activation to upsample and reconstruct the image. To bridge the encoder and decoder, a ConvLSTM is introduced in the latent space, allowing the network to propagate contextual information across different scales. This design avoids the redundancy and overfitting associated with scale-specific CNNs and significantly reduces the number of trainable parameters while maintaining performance.

The network architecture is illustrated in the (fig.16).

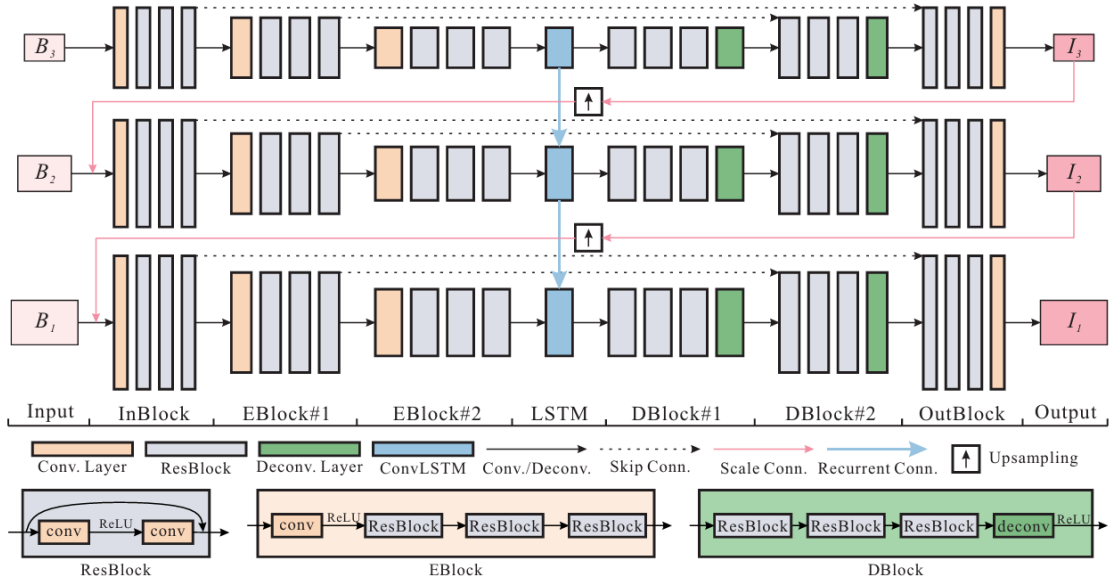


Figure 16: SRN-DeblurNet framework

The loss function is using L_2 to compute at multiple scales. The loss encourages the output image at each scale to be close to the corresponding ground truth sharp image. The formulation is as follows (eq.17):

$$\mathcal{L} = \sum_{i=1}^n \frac{\kappa_i}{N_i} \|I^i - I_*^i\|_2^2, \quad (17)$$

where I^i and I_*^i are our network output and ground truth respectively in the i^{th} scale. κ_i are the weights for each scale. We empirically set $\kappa_i = 1.0$. N_i is the number of elements in I_i to normalize.

Quantitative results

Method	GOPRO		Köhler Dataset		Time
	PSNR	SSIM	PSNR	MSSIM	
Kim <i>et al.</i>	23.64	0.8239	24.68	0.7937	1 hr
Sun <i>et al.</i>	24.64	0.8429	25.22	0.7735	20 min
Nah <i>et al.</i>	29.08	0.9135	26.48	0.8079	3.09 s
Ours	30.26	0.9342	26.75	0.8370	1.87s

Figure 17: Quantitative results

Qualitative results

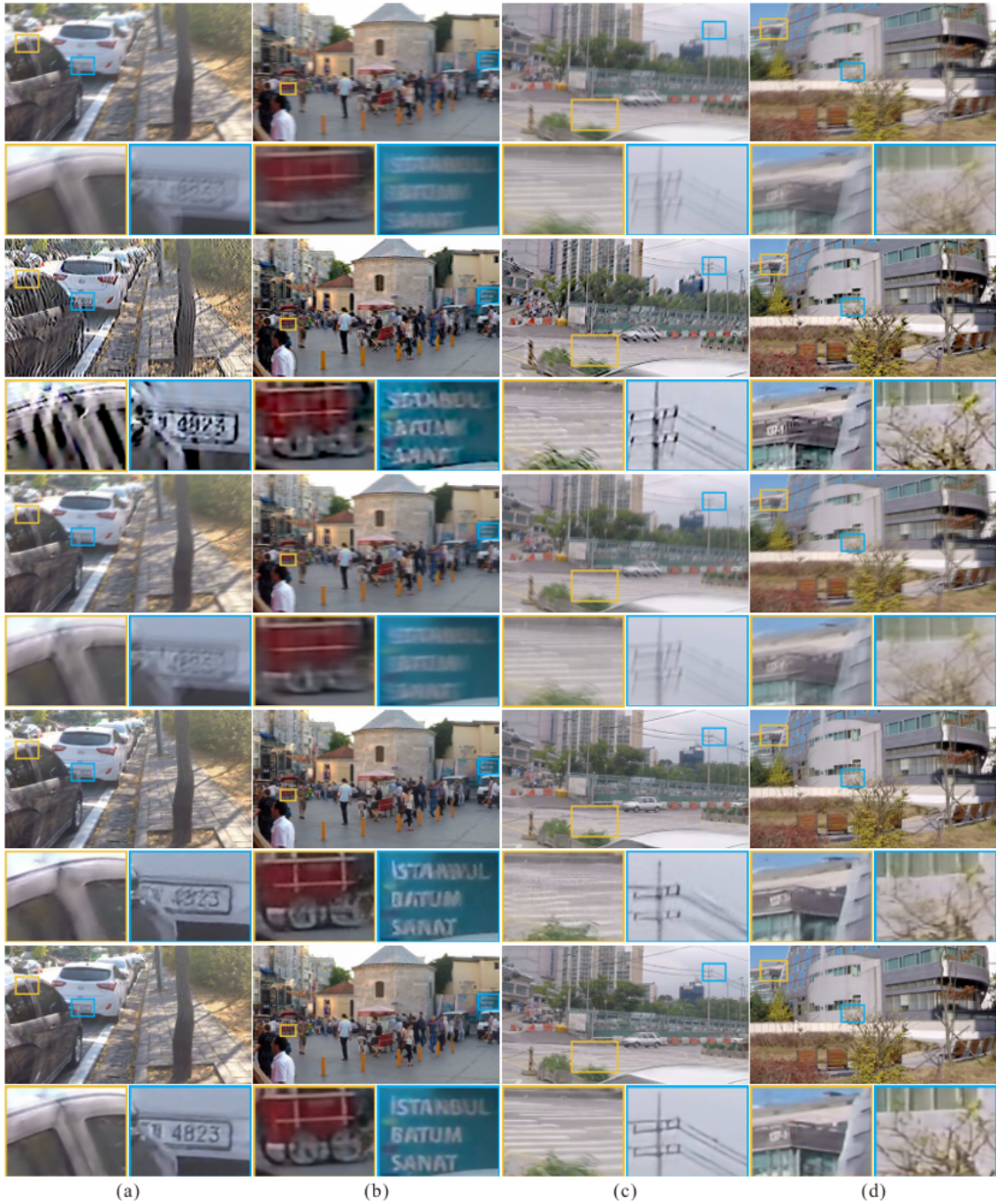


Figure 18: Qualitative results. In the top-down order, we show input, results of Whyte et al., Sun et al., and Nah et al., and our results.

2.2.3 DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks [8]

This paper adopts a Generative Adversarial Network (GAN) framework to restore fine textures and details in deblurred images. The network is trained using a multi-component loss function, combining adversarial and perceptual (content) objectives to enhance both realism and fidelity. Specifically, the adversarial loss is formulated based on the Wasserstein GAN with Gradient Penalty (WGAN-GP), which improves training stability and mitigates mode collapse issues commonly observed in traditional GANs. The content loss is defined as the feature map difference between the generated image and the ground truth, extracted using a pre-trained VGG-19 network. This perceptual loss encourages the restoration of high-level semantic structures and fine textures beyond pixel-wise similarity.

In addition, the paper introduces a motion blur kernel generation strategy as shown in (fig.20), which simulates realistic motion patterns to provide better supervision or data augmentation during training. This helps the network generalize to a wider range of blur types and motion trajectories.

There are three key contributions:

- A multi-component loss function integrated with a generative adversarial network (GAN) is proposed, combining perceptual and adversarial supervision to restore realistic textures and high-frequency details
- A blur synthesis framework based on random motion trajectories is introduced. Using a Markov process, the method simulates plausible object/camera movements to generate motion blur point spread functions (PSFs), which are used to synthesize realistic blurry images.
- A novel evaluation protocol based on object detection performance is proposed to assess the effectiveness of deblurring algorithms. By measuring detection accuracy on deblurred images, the method provides a high-level, task-driven metric that reflects real-world utility beyond pixel-based comparisons.

The overall network architecture is illustrated in the following (fig.19):

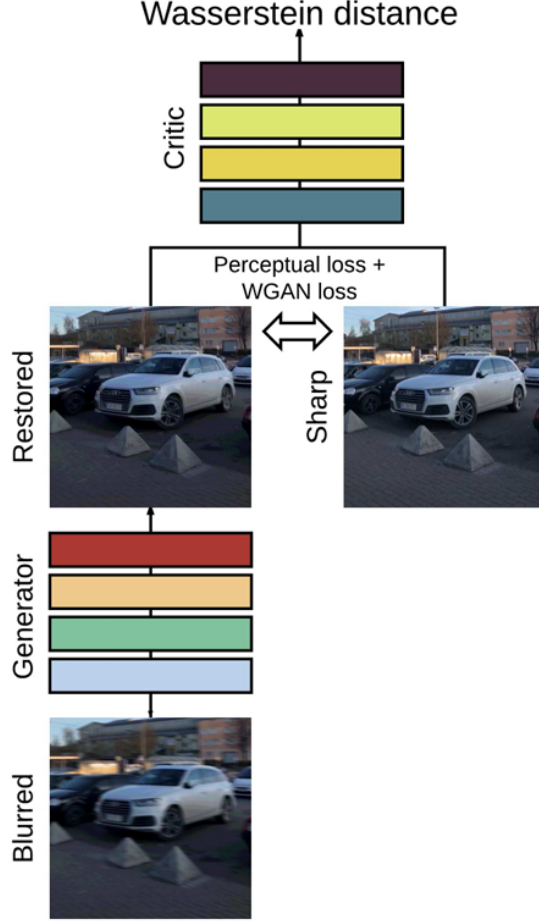


Figure 19: DeblurGAN training. The generator network takes the blurred image as input and produces the estimate of the sharp image. The critic network takes the restored and sharp images and outputs a distance between them. The total loss consists of the WGAN loss from critic and the perceptual loss. The perceptual loss is the difference between the VGG-19 conv3.3 feature maps of the sharp and restored images. At test time, only the generator is kept.

The overall loss function is formulated as a combination of adversarial loss and content (perceptual) loss. The total loss is defined as (eq.18):

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda \mathcal{L}_X, \quad (18)$$

where λ is hyperparameter, $\mathcal{L}_{GAN}$ and $\mathcal{L}_X$ denote the adversarial loss and content loss. for adversarial loss is defined as (eq.19):

$$\mathcal{L}_{GAN} = \sum_{n=1}^N -D_{\theta_D}(G_{\theta_G}(I^B)), \quad (19)$$

where G_{θ_G} is Generator, D_{θ_D} is discriminator. For content loss is defined as (eq.20) :

$$L_X = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^S)_{x,y} - \phi_{i,j}(G_{\theta_G}(I^B))_{x,y})^2, \quad (20)$$

where $\phi_{i,j}$ is the feature map obtained by the j^{th} convolution (after activation) before the i^{th} maxpooling layer within the VGG19 network, pretrained on ImageNet, $W_{i,j}$ and $H_{i,j}$ are the dimensions of the feature maps.

To generate realistic blurry images for training, the authors simulate motion blur using random trajectories generated via a Markov process as show in (fig.20). These trajectories are then

used to construct point spread functions (PSFs), which are applied to sharp images to synthesize corresponding blurry counterparts.

Algorithm 1 Motion blur kernel generation.

Parameters:

$M = 2000$ – number of iterations,

$L_{max} = 60$ – max length of the movement,

$p_s = 0.001$ – probability of impulsive shake,

I – inertia term, uniform from (0,0.7),

p_b – probability of big shake, uniform from (0,0.2),

p_g – probability of gaussian shake, uniform from (0,0.7),

ϕ – initial angle, uniform from (0, 2π),

x – trajectory vector.

```

1: procedure BLUR(Img,  $M$ ,  $L_{max}$ ,  $p_s$ )
2:    $v_0 \leftarrow \cos(\phi) + \sin(\phi) * i$ 
3:    $v \leftarrow v_0 * L_{max} / (M - 1)$ 
4:    $x = \text{zeros}(M, 1)$ 
5:   for  $t = 1$  to  $M - 1$  do
6:     if randn <  $p_b * p_s$  then
7:       nextDir  $\leftarrow 2 * v * e^{i * (\pi + (\text{randn} - 0.5))}$ 
8:     else:
9:       nextDir  $\leftarrow 0$ 
10:     $dv \leftarrow \text{nextDir} + p_s * (p_g * (\text{randn} + i * \text{randn}) * I * x[t] * (L_{max} / (M - 1)))$ 
11:     $v \leftarrow v + dv$ 
12:     $v \leftarrow (v / \text{abs}(v)) * L_{max} / (M - 1)$ 
13:     $x[t + 1] \leftarrow x[t] + v$ 
14:  Kernel  $\leftarrow$  sub pixel interpolation( $x$ )
15:  Blurred image  $\leftarrow \text{conv}(\text{Kernel}, \text{Img})$ 
16:  return Blurred image

```

Figure 20: Motion blur kernel generation.

Quantitative results

	Sun <i>et al.</i>	Nah <i>et al.</i>	Xu <i>et al.</i>	DeblurGAN		
Metric	[36]	[25]	[44]	<i>WILD</i>	<i>Synth</i>	<i>Comb</i>
PSNR	24.6	28.3/ 29.1*	25.1	27.2	23.6	28.7
SSIM	0.842	0.916	0.89	0.954	0.884	0.958
Time	20 min	4.33 s	13.41 s	0.85 s		

Figure 21: Quantitative results on GoPro dataset.

Method	Sun <i>et al.</i>	Nah <i>et al.</i>	Xu <i>et al.</i>	Whyte <i>et al.</i>	DeblurGAN		
Metric	[36]	[25]	[44]	[40]	<i>WILD</i>	<i>Synth</i>	<i>Comb</i>
PSNR	25.22	26.48	27.47	27.03	26.10	25.67	25.86
SSIM	0.773	0.807	0.811	0.809	0.816	0.792	0.802

Figure 22: Quantitative results on Kohler dataset.

Qualitative results

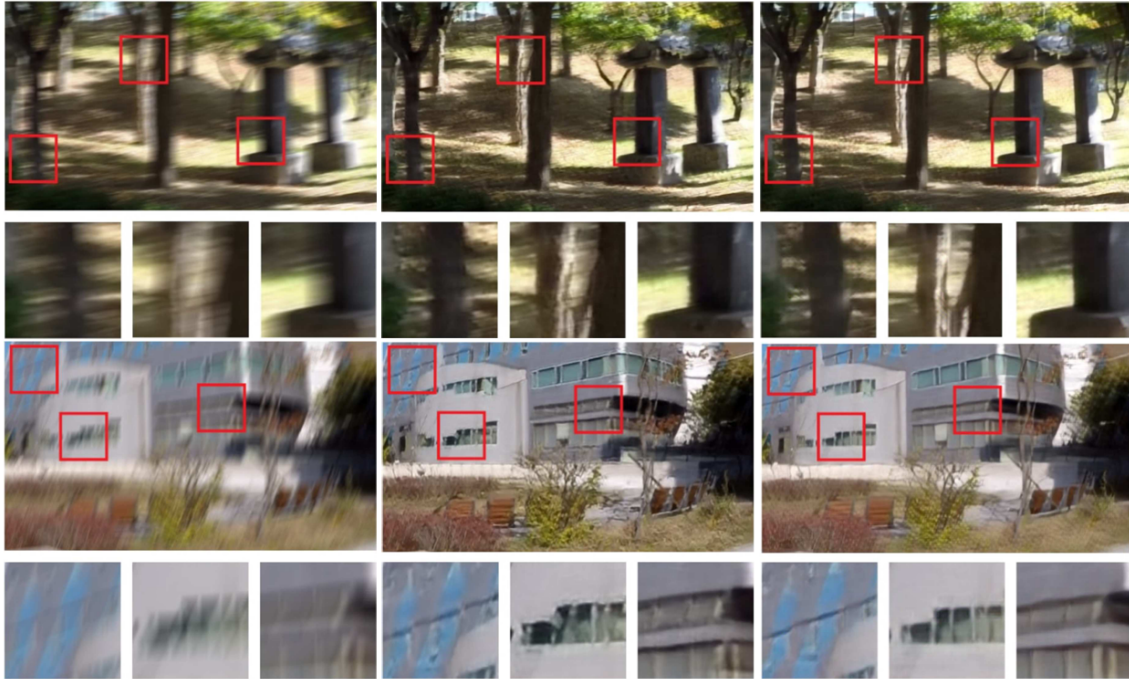


Figure 23: Results on the GoPro test dataset. From left to right: blurred photo, Nah et al., DeblurGAN.



Figure 24: Results on the Kohler dataset. From left to right: blurred photo, Nah et al., DeblurGAN.

Furthermore, to evaluate the effectiveness of the deblurring model in a practical high-level vision task, the authors employ YOLO, a state-of-the-art object detection framework, to compare performance on both blurry and deblurred images. As illustrated in (fig.25) and (fig.26), the object detection accuracy significantly improves after applying the proposed deblurring method. This result demonstrates that the restored images not only look visually sharper but also contain more recoverable semantic information, which directly benefits downstream vision tasks. Such improvements further underscore the practical value of the proposed approach beyond traditional low-level metrics.

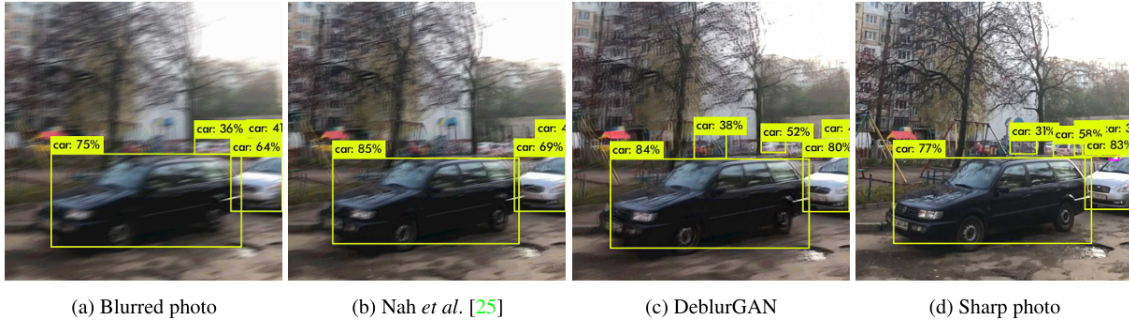


Figure 25: YOLO object detection before and after deblurring.

Method	prec.	recall	F1 score
no deblur	0.821	0.437	0.570
Nah et al. [25]	0.834	0.552	0.665
DeblurGAN WILD	0.764	0.631	0.691
DeblurGAN synth	0.801	0.517	0.628
DeblurGAN comb	0.671	0.742	0.704

Figure 26: Results of YOLO object detection on blurred and restored photos using DeblurGAN and Nah et al. algorithms. Results on corresponding sharp images are considered ground truth. DeblurGAN has higher recall and F1 score than its competitors.

2.2.4 DeblurGAN-v2: Deblurring (Orders-of-Magnitude) Faster and Better [9]

This paper proposes a GAN-based framework for image deblurring by incorporating a Feature Pyramid Network (FPN) into the generator architecture, which facilitates multi-scale feature aggregation to better handle blur at different resolutions. The framework is modular and flexible, allowing various pre-trained backbones—such as Inception-ResNet-v2, MobileNet, and others—to be integrated as the feature extractor. To improve training stability, the authors adopt Relativistic Average GAN with Least Squares loss (RaGAN-LS). Unlike standard GANs, RaGAN-LS encourages the discriminator to estimate the relative realness between real and generated samples, leading to smoother training dynamics and higher-quality outputs. The discriminator design includes dual-scale branches, enabling it to simultaneously capture global structures and local texture details. This dual-scale mechanism ensures that both coarse image consistency and fine-grained patterns are preserved during adversarial training.

Additionally, to enhance the visual quality of the training dataset, the authors employ an interpolation-based frame synthesis strategy, which improves motion smoothness and temporal consistency in cases where blur is caused by frame accumulation (e.g., in video-derived datasets).

The network architecture is illustrated in the following (fig.27):

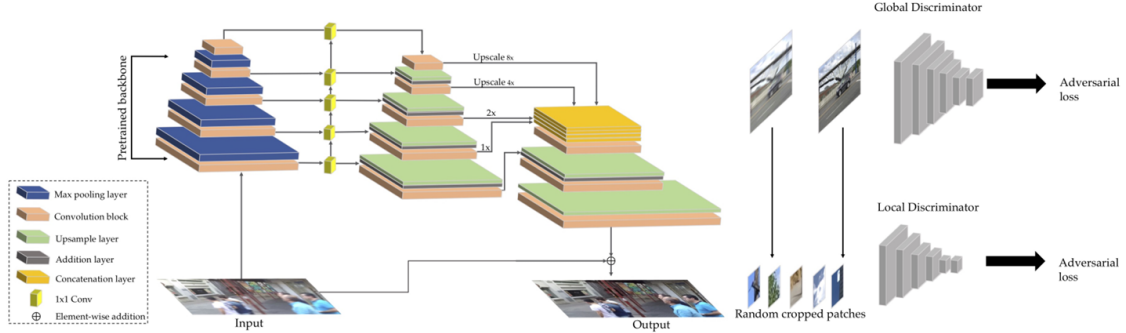


Figure 27: DeblurGAN-v2 pipeline architecture.

There are three main contributions:

- Integration of a Feature Pyramid Network (FPN) architecture into the generator, combined with a relativistic discriminator (RaGAN-LS) to improve multi-scale detail restoration and adversarial training stability.
- The proposed framework supports pluggable backbone networks, allowing different pre-trained models (e.g., Inception-ResNet-v2, MobileNet) to be used depending on the target application. This provides a flexible trade-off between deblurring quality and computational efficiency.
- A high-quality blur dataset generation strategy is proposed using interpolation-based techniques, resulting in improved visual realism and more effective training supervision.

Dataset enhancement via video frame interpolation is applied. To generate high-quality synthetic blur for training, the authors adopt a video frame interpolation model to densify the temporal resolution of videos. This allows for more intermediate sharp frames to be averaged during blur synthesis. Reduced ghosting artifacts, resulting in more realistic blurry images and improved training supervision. A figure illustrating the blur synthesis is shown in the (fig.28) below.

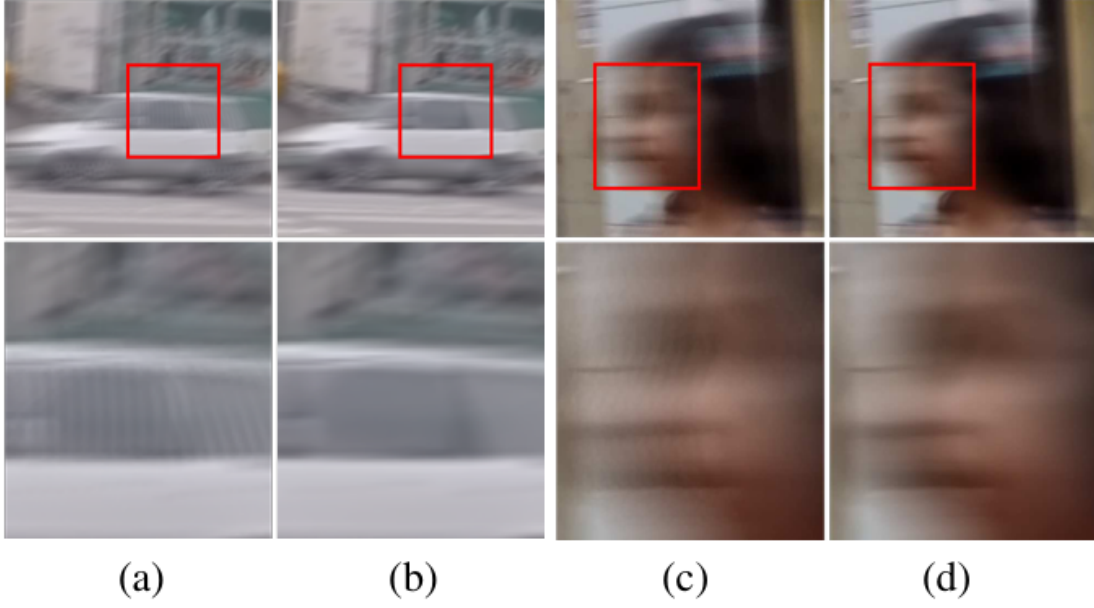


Figure 28: Visual comparison of synthesized blurry images, without interpolation (a,c) and with interpolation (b,d).

Loss function is combine with three components content loss, VGG19 feature map loss and adversarial loss. It defined by follow (eq.21)

$$L_G = 0.5 * L_p + 0.006 * L_X + 0.01 * L_{adv}, \quad (21)$$

where L_{adv} defined as

$$L_D^{RaLSGAN} = \mathbb{E}_{x \sim P_{data}(x)} [(D(x) - \mathbb{E}_{z \sim p_z(z)} D(G(z)) - 1)^2] + \mathbb{E}_{z \sim P_z(z)} [(D(G(z)) - \mathbb{E}_{x \sim p_{data}(x)} D(x) + 1)^2], \quad (22)$$

L_p denotes L_1 or L_2 distance.

L_X denotes VGG19 feature map loss.

Quantitative results

	Sun <i>et al.</i> [43]	Xu <i>et al.</i> [51]	DeepDeblur [33]	SRN [45]	DeblurGAN [21]	Inception-ResNet-v2	MobileNet	MobileNet-DSC
PSNR	24.64	25.10	29.23	30.10	28.70	29.55	28.17	28.03
SSIM	0.842	0.890	0.916	0.932	0.927	0.934	0.925	0.922
Time	20 min	13.41s	4.33s	1.6s	0.85s	0.35s	0.06s	0.04s
FLOPS	N/A	N/A	1760.04G	1434.82G	678.29G	411.34G	43.75G	14.83G

Figure 29: Quantitative results on the GoPro dataset.

Method	Sun [43]	DeepDeblur [33]	SRN [45]	DeblurGAN [21]	Inception-ResNet-v2	MobileNet	MobileNet-DSC
PSNR	25.22	26.48	26.75	26.10	26.72	26.36	26.35
SSIM	0.773	0.807	0.837	0.816	0.836	0.820	0.819

Figure 30: Quantitative results on the Kohler dataset.

Qualitative results

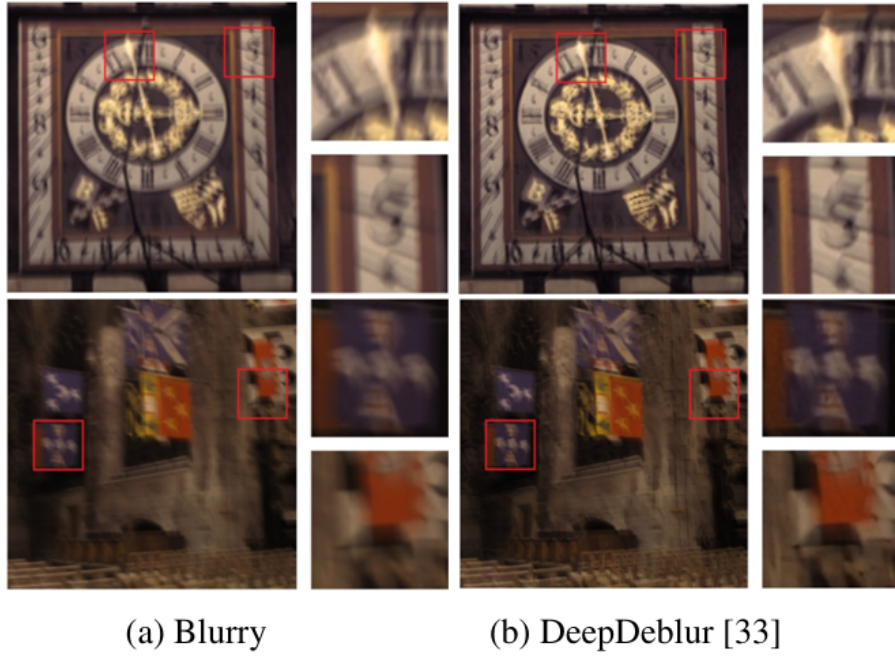


Figure 31: Qualitative results on the Kohler dataset



Figure 32: Qualitative results on the Kohler dataset



(e) DeblurGAN-v2
(Inception-ResNet-v2)

(f) DeblurGAN-v2
(MobileNet)

Figure 33: Qualitative results on the Kohler dataset

3 UNet-based End-to-End Architectures

Most UNet-based image deblurring methods can be broadly categorized into two approaches: (1) Multi-stage architectures that progressively perform deblurring in a stepwise manner, refining the image through successive stages. (2) Block-level improvements within the UNet backbone, where researchers enhance the building blocks by integrating advanced components such as self-attention mechanisms, MLP modules, or state space models (SSMs) to better capture long-range dependencies and complex blur patterns. In the following subsections, we provide a detailed review of representative papers.

3.1 CNN-Single stage

3.1.1 Rethinking Coarse-to-Fine Approach in Single Image Deblurring [4]

Inspired by DeepDeblur, which demonstrated the effectiveness of coarse-to-fine strategies, the authors observe that such architectures may suffer from efficiency bottlenecks due to their large model size and computational cost. To address this, the proposed method retains the coarse-to-fine strategy but re-designs the architecture using a more compact and effective UNet-based multi-scale framework, termed MIMO-UNet (Multi-Input Multi-Output UNet).

In this design, image restoration is performed at each scale using residual learning, allowing intermediate supervision and progressive refinement. The overall architecture introduces several key improvements over the standard UNet and is divided into three main components: (1) Multi-Input Single Encoder (MISE): Aggregates information from multiple scales at the input level and encodes it into a shared representation. (2) Asymmetric Feature Fusion (AFF): Enhances information exchange between encoder and decoder by fusing features of different resolutions in an asymmetric manner, improving both semantic consistency and spatial detail. (3) Multi-Output Single Decoder (MOSD): Produces restored outputs at multiple resolutions through a shared decoder pathway, enabling coarse-to-fine refinement while maintaining architectural compactness. The network of the proposed MIMO-UNet is shown in the (fig.34) below.

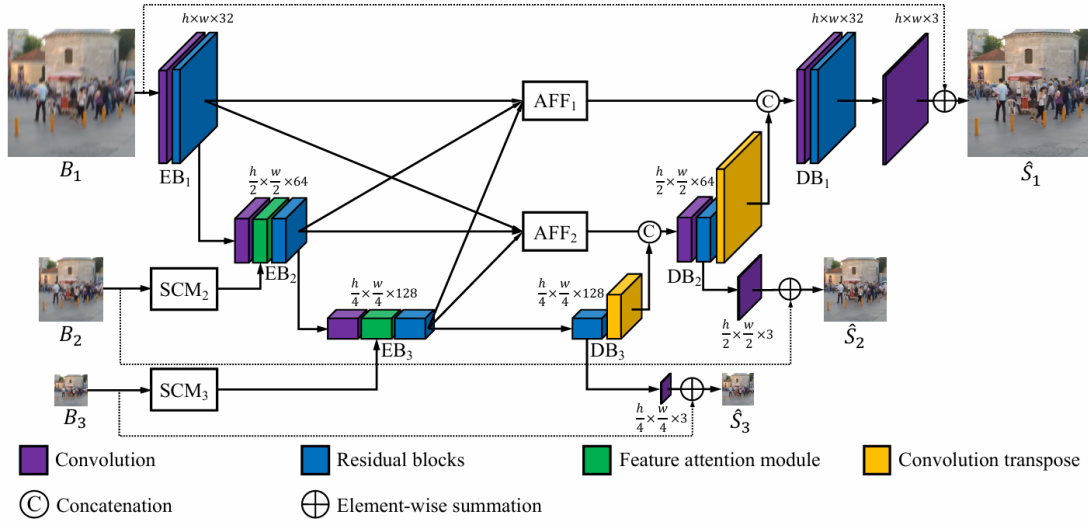


Figure 34: The architecture of the proposed network.

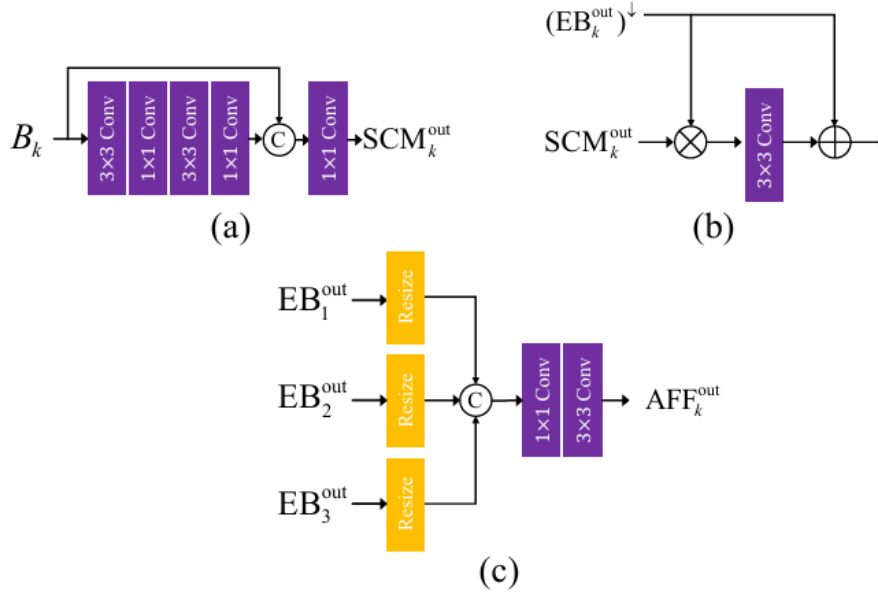


Figure 35: The structures of sub-modules: (a) SCM, (b) feature attention, and (c) AFF.

There are three main contributions of this paper can be summarized as follows:

- **Multi-Scale Input Encoder:** The proposed UNet-based architecture enables the encoder to simultaneously process multi-scale inputs, effectively capturing hierarchical contextual information from different resolutions.
- **Multi-Scale Output Decoder:** A multi-output decoder is introduced to generate restored images at multiple scales, allowing for coarse-to-fine supervision and progressive refinement.
- **Asymmetric Feature Fusion (AFF):** A novel feature fusion strategy is designed to merge encoder and decoder features of different scales in an asymmetric manner. This improves information flow in skip connections and enhances the model’s ability to restore both global structure and fine details.

The proposed MIMO-UNet enhances the classic UNet structure by integrating three key components shown in (fig.35): the Shallow Convolutional Module (SCM), Feature Attention Module (FAM), and Asymmetric Feature Fusion (AFF), all within a coarse-to-fine multi-scale learning framework. Below is a detailed explanation of each module and its role in the encoder-decoder pipeline.

In the encoder, each Encoder Block (EB) begins with a convolutional layer followed by a residual block to extract features. All EB layers (except the first) are enhanced with a Feature Attention Module (FAM) to incorporate multi-scale attention-guided context. The first EB omits the SCM and FAM and directly encodes the input features. Intermediate encoder blocks include: (1) A Shallow Convolutional Module (SCM) to process the downsampled blurry image. The SCM is designed to extract shallow features from downsampled blurry images and is used in all encoder layers except the first. Its structure consists of: Two stacked convolutional layers: one 3×3 followed by one 1×1 . The output is concatenated with the downsampled blurry image. A final 1×1 convolution refines the fused representation. (2) A FAM to merge attention-weighted features from both the previous EB and the corresponding SCM output. Inserted between the convolutional layer and residual block in each EB (except the first), the FAM selectively emphasizes or suppresses information from the previous scale’s EB output and the current SCM output. The process is as follows: The SCM output and the previous EB output are element-wise multiplied (similar to applying an attention map). The result is passed through a 3×3 convolution. The output is then added back to the current EB feature to form the final residual input.

Each Decoder Block (DB) begins with a convolutional layer followed by a residual block, except the bottom-most DB, which does not include the initial convolution. The decoder progressively upsamples and reconstructs the image features, with multiple outputs generated at different resolutions to support coarse-to-fine learning.

To replace the traditional skip connections in UNet, the authors propose Asymmetric Feature Fusion (AFF) to fuse multi-scale features effectively. Features from all encoder blocks are resized to match the target resolution (e.g., 256×256 for the first output). The resized features are concatenated, followed by: A 1×1 convolution to reduce channel dimensions. A 3×3 convolution to spatially integrate the fused features. AFF ensures that the decoder receives context-rich information aggregated from all encoder levels, enhancing the model’s ability to reconstruct sharp images across scales.

Loss function is combined with Content loss and Multi-scale frequency reconstruction (MSFR) loss. It is defined as (eq.23):

$$L_{total} = L_{cont} + \lambda L_{MSFR}, \quad (23)$$

where λ is hyperparameter, L_{cont} and L_{MSFR} denote the content loss and multi-scale frequency reconstruction loss. For Content loss (eq.24):

$$L_{cont} = \sum_{k=1}^K \frac{1}{t_k} \|\hat{S}_k - S_k\|_1, \quad (24)$$

where K is the number of level, t_k is number of total elements for normalization, $\hat{S}_k$ and S_k denotes deblurring images and ground truth. For MSFR loss (eq.25):

$$L_{MSFR} = \sum_{k=1}^K \frac{1}{t_k} \|\mathcal{F}(\hat{S}_k) - \mathcal{F}(S_k)\|_1, \quad (25)$$

where $\mathcal{F}(\cdot)$ is FFT.

Quantitative results

Model	PSNR	SSIM	Runtime		Params.
DeepDeblur [20]	29.23	0.916	N/A	4.33	11.7
SRN [31]	30.26	0.934	0.342	1.87	6.8
PSS-NSC [5]	30.92	0.942	0.985	1.6	<u>2.84</u>
DMPHN [35]	31.20	0.945	1.061	0.424	21.7
SAPHN [†] [29]	31.85	0.948	N/A	0.34	N/A
SAPHN [‡] [29]	32.02	0.953	N/A	0.77	N/A
MT-RNN [22]	31.15	0.945	0.063	0.07	2.6
RADN [23]	31.76	0.953	N/A	0.038	N/A
SVDN [33]	29.81	0.937	N/A	<u>0.01</u>	N/A
MPRNet [34]	<u>32.66</u>	0.959	0.162	0.18	20.1
MIMO-UNet	31.73	0.951	0.008		6.8
MIMO-UNet+	32.45	0.957	0.017		16.1
MIMO-UNet++	32.68	0.959	0.040		16.1

Figure 36: Quantitative results on GoPro dataset. There are different version MIMO-UNet that is the number of residual blocks in encoder or decoder. MIMO-UNet: 8 residual blocks. MIMO-UNet+: 20 residual blocks. MIMO-UNet++: MIMO-UNet+ with geometric self-ensemble.

Model	PSNR	SSIM
DeblurGAN-v2 [15]	29.69	0.870
SRN [31]	31.38	0.909
MPRNet [34]	31.76	0.922
MIMO-UNet+	<u>31.92</u>	0.919
MIMO-UNet++	32.05	<u>0.921</u>

Figure 37: Quantitative results on RealBlur dataset.

Qualitative results



Figure 38: Qualitative results on the GoPro test dataset. For clarity, the magnified parts of the resultant images are displayed. From left-top to right-bottom: Blurry images, ground-truth images, and the resultant images obtained by SRN, PSS-NSC, DMPHN, MT-RNN, MPRNet, and MIMO-UNet++, respectively.

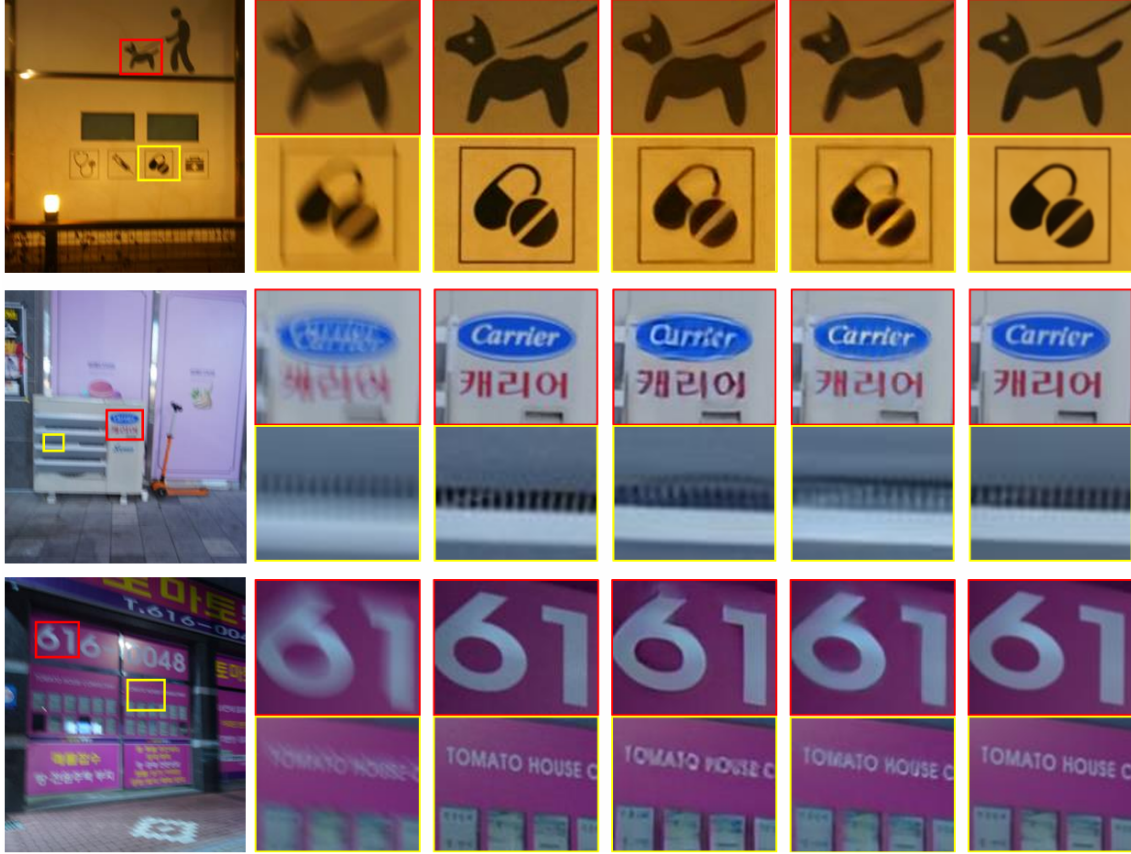


Figure 39: Qualitative results on the RealBlur test dataset. For clarity, the magnified parts of the resultant images are displayed. From left to right: Blurry images, ground-truth images, and the resultant images obtained by DeblurGAN-v2, SRN, and MIMO-UNet++, respectively.

3.1.2 Simple Baselines for Image Restoration [1]

This paper focuses on analyzing and simplifying both the inter-block and intra-block complexity of UNet-based deblurring architectures: (1) Inter-block complexity refers to the macro-level design of the overall architecture, such as MIMO-UNet, SISO-UNet, and multi-stage frameworks as illustrated in the (fig.40). (2) Intra-block complexity focuses on the design within each block, including convolutional structures, attention mechanisms, and activation functions as illustrated in the (fig.41).

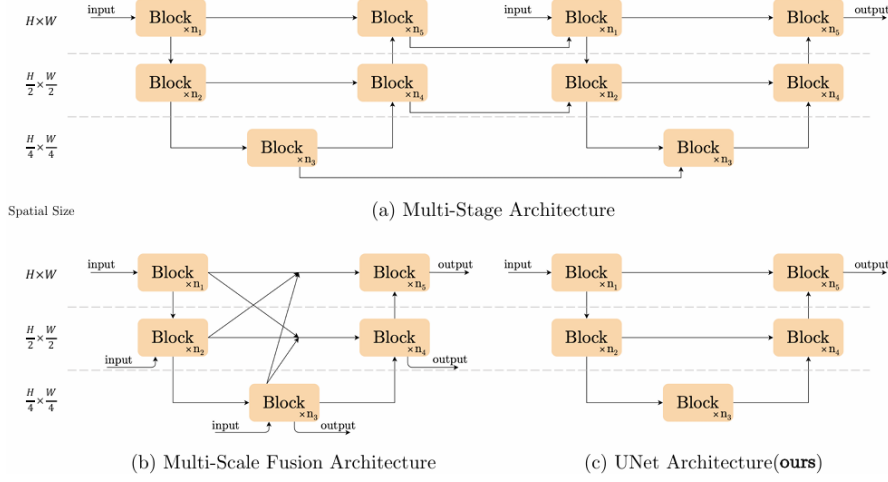


Figure 40: Comparison of architectures of image restoration models. Dashes to distinguish features of different sizes. (a) The multi-stage architecture stacks UNet architecture serially. (b) The multi-scale fusion architecture fuses the features in different scales. (c) UNet architecture, which is adopted by some SOTA methods. We use it as our architecture. Some details have been deliberately omitted for simplicity, e.g. downsample/upsample layers, feature fusion modules, input/output shortcut, and etc.

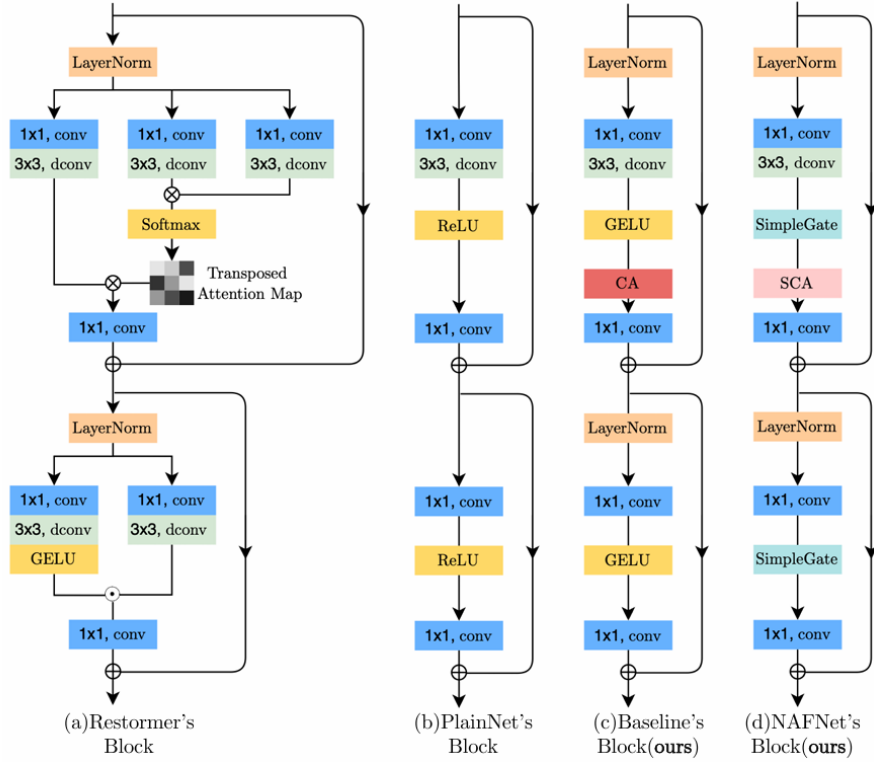


Figure 41: Intra-block structure comparison. $\otimes$:matrix multiplication, $\odot/\oplus$:elementwise multiplication/addition. dconv: Depthwise convolution. Nonlinear activation functions are represented by yellow boxes. (a) Restormer's block, some details are omitted for simplicity, e.g. reshaping the feature maps. (b) PlainNet's block, which contains the most common components. (c) Our proposed baseline. Compares to (b), Channel Attention (CA) and LayerNorm are adopted. Besides, ReLU is replaced by GELU. (d) Our proposed Nonlinear Activation Free Network's block. It replaces CA/GELU with Simplified Channel Attention(SCA) and SimpleGate respectively. The details of these components are shown in (fig.fig:3.1.2/intra block detail)

To explore the design space and reduce architectural redundancy, the authors propose two networks:

Baseline Model: A CNN-based SISO-UNet architecture that simplifies inter-block and intra-block complexity by adopting a Single-Input Single-Output structure with channel attention mechanisms. Various components from state-of-the-art deblurring models are evaluated within a Plain-Net backbone, leading to the design of a lightweight, effective baseline model.

Nonlinear activation free network: Building upon the baseline, the authors further propose a nonlinear activation-free network, which eliminates common nonlinearities (e.g., ReLU, GELU) to reduce computational overhead while retaining deblurring performance. This model demonstrates that, under careful design, activation layers can be safely omitted in certain scenarios, yielding a highly efficient architecture.

The main contributions of this work are as follows:

- **Component-wise Decomposition of Residual Blocks:** The authors systematically decompose common components within residual blocks—including normalization layers, nonlinear activation functions, and attention mechanisms—to construct a simplified yet effective baseline model for image deblurring. This modular analysis provides deeper insights into the role of each component in performance and efficiency.
- **Activation-Free Attention Design:** A novel method is proposed to simplify Gated Linear Units (GLU) and Channel Attention (CA) by removing nonlinear activation functions, significantly reducing computational complexity while preserving or even improving deblurring performance. This results in a more lightweight and hardware-friendly network architecture.

In the proposed **baseline model**, the authors argue that a simple UNet architecture is sufficient for effective deblurring. Unlike more complex inter-block designs such as MIMO-UNet or multi-stage models as shown in (fig.40a) and (fig.40b), the baseline adopts a Single-Input Single-Output (SISO) strategy to significantly reduce inter-block complexity while maintaining competitive performance.

A central focus is placed on refining the intra-block structure, where the authors systematically evaluate and simplify the design components based on the general residual block pattern shown in (fig.41b), leading to a more efficient variant (fig.41c). The simplification targets three major components: Normalization, Activation, and Attention.

Normalization While Batch Normalization (BN) is widely used, it performs inconsistently with small batch sizes due to statistical instability. Instance Normalization (IN) can mitigate this but often requires manual tuning and may not consistently boost performance. The authors find that Layer Normalization (LN) provides stable training dynamics and consistent performance improvements, and thus adopt LN throughout the model.

Activation Recent trends have shown a transition from ReLU to GELU activations due to its smoother behavior and superior performance in several vision tasks. The baseline model therefore uses GELU as the activation function, which is shown to be more effective than traditional alternatives.

Attention Standard self-attention mechanisms introduce quadratic computational complexity, making them less suitable for lightweight models. Though window-based attention reduces complexity, it limits global receptive fields. Instead, the authors employ a CNN-based Channel Attention mechanism, which strikes a balance between efficiency and the ability to capture global context, without the overhead of full self-attention.

With these design choices—Layer Normalization, GELU, and Channel Attention—the baseline block achieves state-of-the-art (SOTA) performance while maintaining a relatively low computational cost.

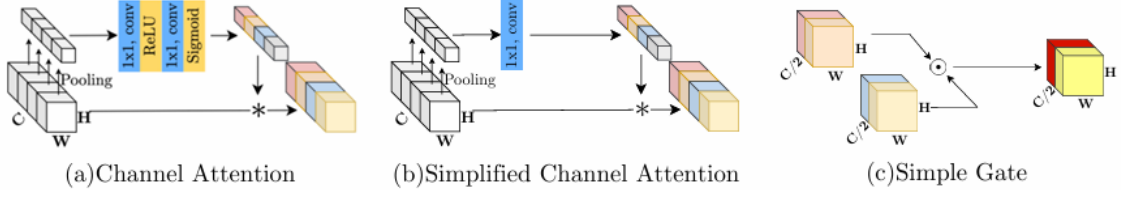


Figure 42: Illustration of (a) Channel Attention (CA), (b) Simplified Channel Attention (SCA), and (c) Simple Gate (SG). $\odot$ / $*$: element-wise/channel-wise multiplication

Building on the baseline model, the authors introduce **Nonlinear Activation-Free Network (NAFNet)**, a further simplified network that removes nonlinear activation functions altogether, while preserving high performance. This is motivated by a set of mathematical insights into the structure of Gated Linear Units (GLU) and attention modules. A standard GLU can be expressed as (eq.26):

$$Gate(X, f, g, \sigma) = f(X) \odot \sigma(g(X)), \quad (26)$$

where $\sigma(\cdot)$ is a nonlinear activation function. Similarly, the GELU activation used in the baseline can be approximated as (eq.27):

$$GELU(x) = x\Phi(x), \quad (27)$$

where $\Phi(\cdot)$ behaves similarly to $\sigma(\cdot)$. Observing this structure, the authors propose a simplified gate, SimpleGate, which entirely removes nonlinearity (eq.28):

$$SimpleGate(X, Y) = X \odot Y, \quad (28)$$

where X and Y are split feature maps.

Likewise, the original Channel Attention (CA) mechanism is formulated as (eq.29):

$$CA(X) = X * \sigma(W_2 \max(0, W_1 \text{pool}(X))) \quad (29)$$

which can be abstracted as (eq.30):

$$CA(X) = X * \Psi(X) \quad (30)$$

In NAFNet, this is simplified to SCA (Simplified Channel Attention) by removing the nonlinearity (eq.31):

$$SCA(X) = X * W \text{pool}(X) \quad (31)$$

where W is a learned linear projection.

With these modifications, NAFNet eliminates all nonlinear activation functions inside blocks, significantly reducing both runtime and parameter complexity while achieving competitive deblurring results. The final residual block structure is shown in the figure (refer to the ResBlock diagram).

Quantitative results

Method	MIMO-UNet [6]	HINet [4]	MAXIM [30]	Restormer [37]	UFormer [34]	DeepRFT [24]	MPRNet -local [7]	Baseline ours	NAFNet ours
PSNR	32.68	32.71	32.86	32.92	32.97	33.23	33.31	33.40	33.69
SSIM	0.959	0.959	0.961	0.961	0.967	0.963	0.964	0.965	0.967
MACs(G)	1235	170.7	169.5	140	89.5	187	778.2	65	65

Figure 43: Quantitative results on GoPro dataset.

Qualitative results



Figure 44: Qualitative results on GoPro dataset.

3.1.3 Intriguing Findings of Frequency Selection for Image Deblurring [11]

This paper investigates the behavior of blurry images in the frequency domain after being processed by ReLU activations, and leverages this understanding to design a novel architecture for image deblurring.

The main contributions are as follows:

- **Frequency-Domain Analysis of Blurry Features after ReLU.** The authors provide a mathematical analysis of how ReLU nonlinearity affects blurry image features in the Fourier domain. This insight helps to understand why standard convolutional blocks might struggle to recover high-frequency details from blurry inputs.
- **Res FFT-ReLU Block for Enhanced Deblurring.** Based on the above analysis, the authors propose the Res FFT-ReLU Block, which integrates a parallel FFT-based branch into a standard residual block. This branch processes features in the frequency domain, allowing the network to better capture and restore high-frequency components suppressed by blur. This architecture significantly improves the model’s ability to handle fine textures and sharp transitions during deblurring.

The core motivation of this paper stems from a frequency-domain analysis of ReLU-activated features derived from blurry inputs. Specifically, the authors investigate how the ReLU function transforms feature representations in the Fourier domain. The analysis begins with the following formulation (eq.32):

$$\hat{b} = \mathcal{F}^{-1}(\sigma(\mathcal{F}(b))), \quad (32)$$

where $b = f * g$, with b representing the blurry image, f the sharp image, g the point spread function (PSF), $\mathcal{F}$ the Fourier transform, and $\sigma(\cdot)$ the ReLU activation function. The expression can be decomposed as a linear combination of the blurry image and the inverse transform of its

magnitude spectrum:

$$\begin{aligned}
\hat{b} &= \mathcal{F}^{-1}(\sigma(\mathcal{F}(b))) \\
&= \mathcal{F}^{-1}\left(\frac{\mathcal{F}(b) + |\mathcal{F}(b)|}{2}\right) \\
&= \frac{b}{2} + \frac{1}{2}\mathcal{F}^{-1}(|\mathcal{F}(b)|) \\
&= \frac{b}{2} + \frac{1}{2}\mathcal{F}^{-1}(|\mathcal{F}(g(t) \otimes f(t))|) \\
&= \frac{b}{2} + \frac{1}{2}\mathcal{F}^{-1}(|\mathcal{F}(g(t)) \cdot \mathcal{F}(f(t))|) \\
&= \frac{b}{2} + \frac{1}{2}\mathcal{F}^{-1}(|(1 + \cos(\omega\epsilon) + j\sin(\omega\epsilon)) \cdot \mathcal{F}(f(t))|) \\
&= \frac{b}{2} + \frac{1}{2}\mathcal{F}^{-1}((1 + \cos(\omega\epsilon))|\mathcal{F}(f(t))|) + \frac{1}{2}\mathcal{F}^{-1}(j\sin(\omega\epsilon)\mathcal{F}(f(t))) \\
&= \frac{1}{2}(\sigma(t) + \frac{1}{2}[\sigma(t - \epsilon) + \sigma(t + \epsilon)]) \otimes \mathcal{F}^{-1}(|\mathcal{F}(f(t))|) + \frac{1}{2}\mathcal{F}^{-1}(j\sin(\omega\epsilon)\mathcal{F}(f(t))) + \frac{b}{2}
\end{aligned} \tag{33}$$

where $|\cdot|$ denotes taking the absolute value of both the real and imaginary components separately. The term $(\sigma(t) + \frac{1}{2}[\sigma(t - \epsilon) + \sigma(t + \epsilon)]) \otimes \mathcal{F}^{-1}(|\mathcal{F}(f(t))|)$ in the expression for $\hat{b}$ explicitly encodes the blur pattern. This component originates from the interaction between the ReLU nonlinearity and the frequency-domain representation of the sharp image, indicating that information about the blur kernel—such as its direction and magnitude—is preserved within this formulation. The example is shown on (fig.45)

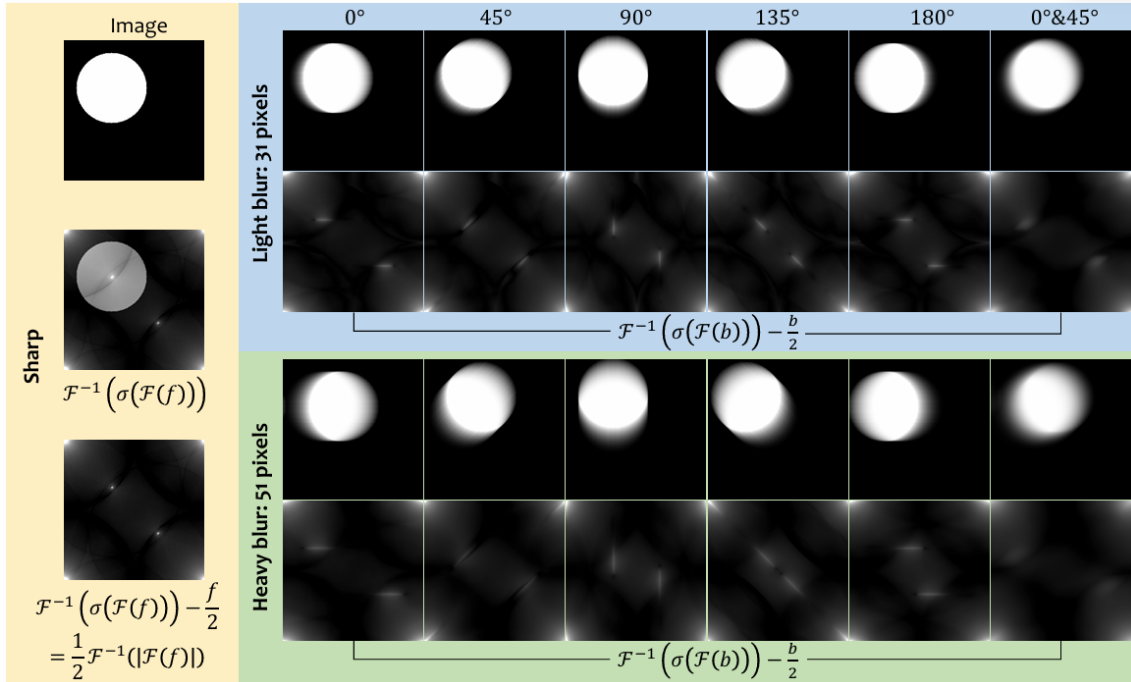


Figure 45: We use a circle image as an example for sharp image f . The image is blurred (b) by linear kernels with motion blur of different directions $\{0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ\}$, different blur levels and a mixture of blur. Blur kernel-like images are calculated by the equations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

Motivated by the previous observations, the authors propose a modified residual block that incorporates an additional FFT-ReLU branch. In this design, after applying the Fast Fourier Transform (FFT), the feature map is decomposed into its amplitude and phase components. The

ReLU activation is applied specifically to the phase, as it is considered more informative and sensitive to blur characteristics in blurry images.

Since the deblurring performance is sensitive to the manually set ReLU threshold, the authors introduce a learnable mechanism by replacing the fixed threshold with a 1×1 convolutional layer, which allows the network to automatically adapt and learn appropriate threshold values during training. The structure of the proposed residual block, termed Res FFT-ReLU Block, is illustrated as follows (fig.46c).

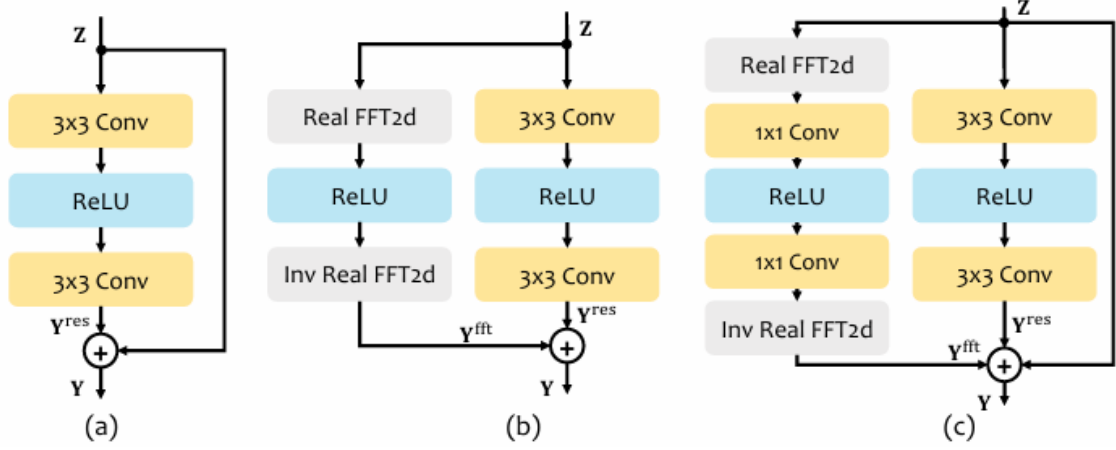


Figure 46: (a) ResBlock. (b) Insert simple FFT-ReLU stream into ResBlock. (c) Proposed Res FFT-ReLU Block, where the bottom stream is termed as FFT-ReLU stream.

The overall loss function consists of the Charbonnier loss ($\mathcal{L}_{char}$) and the Frequency Reconstruction Loss ($\mathcal{L}_{freq}$). The Charbonnier loss is defined as (eq.34):

$$\mathcal{L}_{char} = \sqrt{\|\hat{S} - S\|^2 + \epsilon^2}, \quad (34)$$

where $\hat{S}$ is the predicted sharp image, S is the ground truth sharp image, and $\epsilon = 10^{-3}$ is a small constant to ensure numerical stability. The Frequency Reconstruction Loss is formulated as (eq.35):

$$\mathcal{L}_{freq} = \|\mathcal{F}(\hat{S}) - \mathcal{F}(S)\|_1, \quad (35)$$

where $\mathcal{F}(\cdot)$ denotes the Fast Fourier Transform (FFT), which measures the discrepancy between the predicted and ground truth images in the frequency domain. The final loss function is a weighted combination of the two components, it is defined in (eq.36):

$$\mathcal{L} = \mathcal{L}_{charbonnier} + \lambda \mathcal{L}_{freq}, \quad (36)$$

where $\lambda = 0.01$ is a hyperparameter that balances the contribution of the frequency-domain loss.

Quantitative results

Method	GoPro		HIDE		RealBlur-R		RealBlur-J	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Xu <i>et al.</i> (CVPR'13)	21.00	0.741	-	-	34.46	0.937	27.14	0.830
Hu <i>et al.</i> (CVPR'14)	-	-	-	-	33.67	0.916	26.41	0.803
Pan <i>et al.</i> (CVPR'16)	-	-	-	-	34.01	0.916	27.22	0.790
Nah <i>et al.</i> (CVPR'17)	29.08	0.914	25.73	0.874	32.51	0.841	27.87	0.827
SRN (CVPR'18)	30.26	0.934	28.36	0.915	35.66	0.947	28.56	0.867
Zhang <i>et al.</i> (CVPR'18)	29.19	0.931	-	-	35.48	0.947	27.80	0.847
DeblurGAN (CVPR'18)	28.70	0.858	24.51	0.871	33.79	0.903	27.97	0.834
DeblurGAN-v2 (ICCV'19)	29.55	0.934	26.61	0.875	35.26	0.944	28.70	0.866
Gao <i>et al.</i> (CVPR'19)	30.90	0.935	29.11	0.913	-	-	-	-
DMPHN (CVPR'19)	31.20	0.940	29.09	0.924	35.70	0.948	28.42	0.860
RADN (AAAI'20)	31.76	0.953	-	-	-	-	-	-
Suin <i>et al.</i> (CVPR'20)	31.85	0.948	29.98	0.930	-	-	-	-
SVDN (CVPR'20)	29.81	0.937	-	-	-	-	-	-
DBGAN (CVPR'20)	31.10	0.942	28.94	0.915	33.78	0.909	24.93	0.745
MT-RNN (ECCV'20)	31.15	0.945	29.15	0.918	35.79	0.951	28.44	0.862
MPRNet (CVPR'21)	32.66	0.959	30.96	0.939	35.99	0.952	28.70	0.873
MIMO-UNet (ICCV'21)	31.73	0.951	29.28	0.921	35.47	0.946	27.76	0.836
MIMO-UNet+ (ICCV'21)	32.45	0.957	29.99	0.930	35.54	0.947	27.63	0.837
MIMO-UNet++ (ICCV'21)	32.68	0.959	-	-	-	-	-	-
SPAIR (ICCV'21)	32.06	0.953	30.29	0.931	-	-	28.81	0.875
HINet (CVPRW'21)	32.71	0.959	-	-	-	-	-	-
Restormer (CVPR'22)	32.92	0.961	31.22	0.942	36.19	0.957	28.96	0.879
Uformer (CVPR'22)	33.06	0.967	30.90	0.953	36.19	0.956	29.09	0.886
NAFNet32 (ECCV'22)	32.85	0.959	30.60	0.936	35.97	0.951	28.75	0.875
NAFNet64 (ECCV'22)	33.69	0.967	31.32	0.943	-	-	-	-
FMIMO-UNet-small	32.50	0.958	30.65	0.937	35.92	0.952	28.58	0.864
FMIMO-UNet	33.08	0.962	31.19	0.943	35.96	0.953	28.72	0.871
FMIMO-UNet+	33.52	0.965	31.66	0.946	36.11	0.955	28.88	0.880
FNAFNet32	33.12	0.962	30.76	0.938	36.07	0.955	28.78	0.879
FNAFNet64	33.85	0.967	31.12	0.944	-	-	-	-
DeblurGAN-v2 🚩 (ICCV'19)	-	-	-	-	36.44	0.935	29.69	0.870
SRN 🚩 (CVPR'18)	-	-	-	-	38.65	0.965	31.38	0.909
MPRNet 🚩 (CVPR'21)	-	-	-	-	39.31	0.972	31.76	0.922
FMIMO-UNet 🚩	-	-	-	-	40.01	0.972	32.65	0.931
FMIMO-UNet+ 🚩	-	-	-	-	40.01	0.973	32.63	0.933

Figure 47: Quantitative results on GoPro, HIDE and RealBlur datasets

3.2 CNN-Multi stage

3.2.1 Multi-Stage Progressive Image Restoration [20]

This paper proposes a CNN-based multi-stage architecture for image deblurring. Traditional encoder-decoder architectures are capable of capturing broad contextual features, but the down-sampling and upsampling operations often lead to a loss of spatial detail. In contrast, single-scale pipelines are able to better preserve spatial details but lack sufficient contextual understanding. To balance these two aspects, the authors integrate both architectures: the earlier stages adopt an encoder-decoder design to extract contextual information, while the final stage omits downsampling in order to retain fine spatial details.

To enhance inter-stage communication and supervision, the authors introduce a Supervised Attention Module (SAM), which serves two main purposes: (1) to explicitly supervise the intermediate deblurring results at each stage, and (2) to generate attention maps that suppress irrelevant features before passing them to the next stage. Additionally, a Cross-Stage Feature Fusion (CSFF) mechanism is proposed to effectively transfer hierarchical features across stages, inspired by multi-patch learning networks.

The overall architecture is illustrated as follows (fig.48): the first two stages employ a UNet structure, each followed by a SAM module. The final stage removes the encoder-decoder structure to preserve spatial fidelity, leveraging the information passed from earlier stages via CSFF.

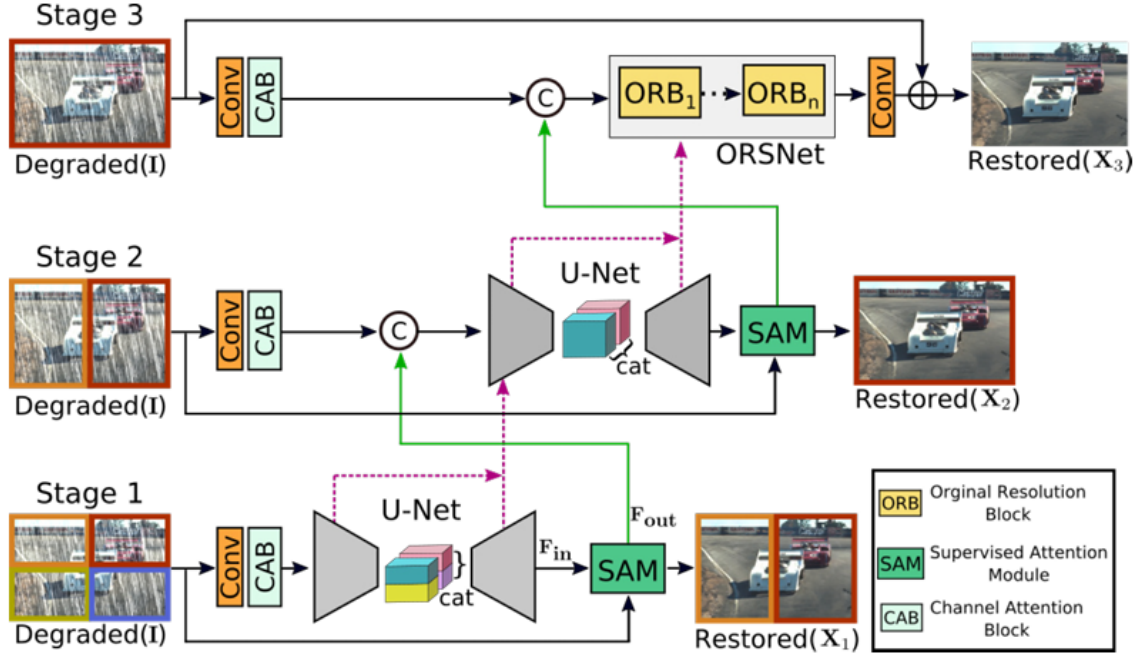


Figure 48: Proposed multi-stage architecture for progressive image restoration. Earlier stages employ encoder-decoders to extract multi-scale contextualized features, while the last stage operates at the original image resolution to generate spatially accurate outputs. A supervised attention module is added between every two stages that learns to refine features of one stage before passing them to the next stage. Dotted pink arrows represent the crossstage feature fusion mechanism.

There are three main contributions:

- Multi-stage architecture that generates contextually enriched and spatially accurate deblurring outputs by combining encoder-decoder and single-scale processing strategies.
- Supervised Attention Module (SAM) that enables effective intermediate deblurring supervision and refines feature transmission to the next stage by emphasizing informative features and suppressing irrelevant ones.

- Cross-Stage Feature Aggregation, which effectively integrates hierarchical features across stages to enhance consistency and boost final restoration performance.

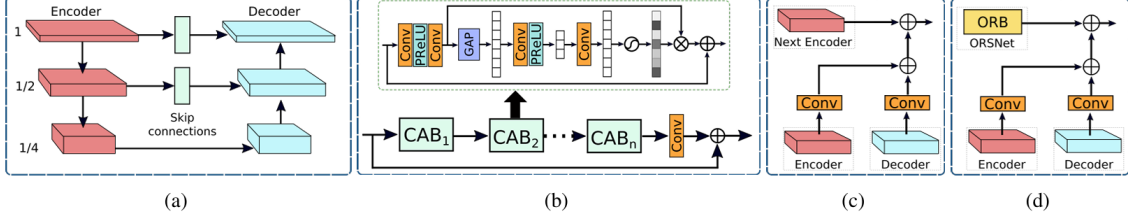


Figure 49: (a) Encoder-decoder subnetwork. (b) Illustration of the original resolution block (ORB) in our ORSNet subnetwork. Each ORB contains multiple channel attention blocks. GAP represents global average pooling. (c) Cross-stage feature fusion between stage 1 and stage 2. (d) CSFF between stage 2 and the last stage.

The proposed method adopts a multi-stage architecture that balances contextual understanding and spatial detail preservation through different sub-networks and modules. The design includes the following components:

1. Encoder-Decoder Subnetwork (fig.49a)

This subnetwork is based on the classical U-Net architecture, which is effective for capturing broad contextual features through repeated downsampling and upsampling. To reduce artifacts commonly introduced by transposed convolution, the model replaces it with bilinear upsampling.

Pros: Large receptive field, strong semantic context

Cons: Spatial detail degradation due to repeated downsampling

2. Original Resolution Subnetwork (fig.49b)

This subnetwork works at the original resolution using a single-scale feed-forward structure without any downsampling, thus maintaining pixel-to-pixel correspondence between the input and output.

Pros: Preserves spatial details

Cons: Limited receptive field, reduced semantic understanding

3. Cross-Stage Feature Fusion (CSFF) (fig.49c) and (fig.49d)

To bridge the gap between stages, CSFF propagates intermediate features from one stage to the next. This module fuses encoder and decoder features from previous stages into the current stage, mitigating information loss from upsampling/downsampling and stabilizing optimization through enhanced feature flow.

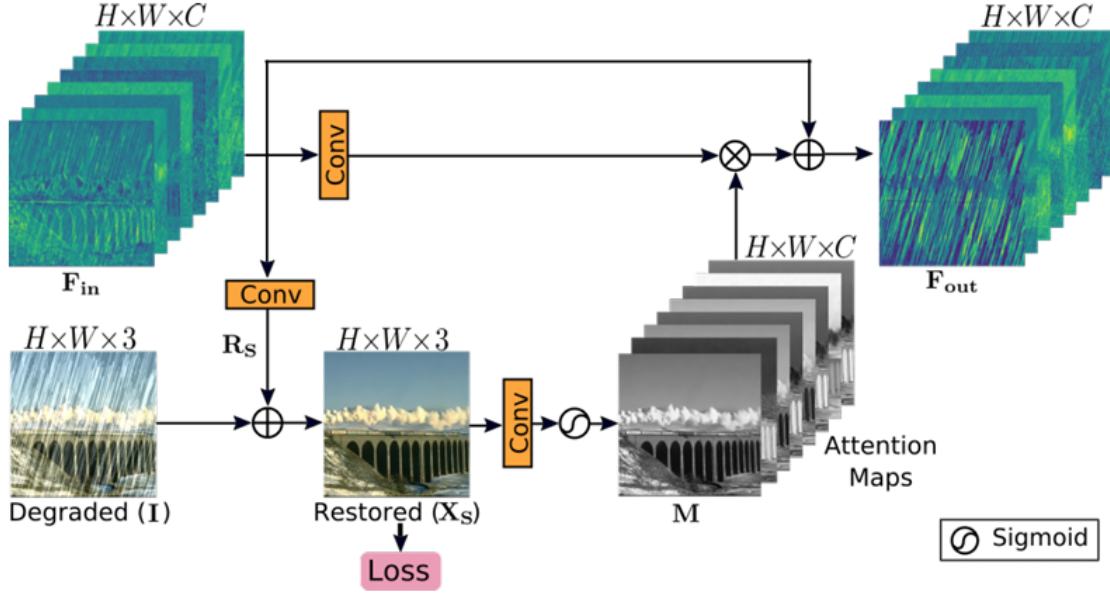


Figure 50: Supervised attention module.

4. Supervised Attention Module (SAM) (fig.50)

SAM is introduced between stages to supervise the intermediate restoration quality using ground truth images. It also generates attention maps from the deblurred output of the current stage to suppress irrelevant information and highlight informative features for the next stage, facilitating effective feature refinement.

5. Multi-Patch Hierarchical Processing Strategy

To mimic a coarse-to-fine strategy similar to Feature Pyramid Networks (FPN), the input blurry image is divided into non-overlapping patches and processed at different stages: Stage 1: 4 patches. Stage 2: 2 patches. Stage 3: Full resolution image. This multi-patch strategy helps capture localized features, reduce boundary artifacts, and enables progressive refinement across scales.

The loss function employed in this work is composed of the sum of the Charbonnier loss ($\mathcal{L}_{char}$) and the Frequency Reconstruction loss ($\mathcal{L}_{freq}$) computed at each stage of the model. The overall loss is formulated as follows (eq.37):

$$\mathcal{L} = \sum_{S=1}^3 [\mathcal{L}_{char}(X_S, Y) + \lambda \mathcal{L}_{edge}(X_S, Y)], \quad (37)$$

where X_S denotes the output of the model at stage S , Y is the ground truth image, and λ is a weighting factor set to 0.05. The Charbonnier loss is defined as (eq.38):

$$\mathcal{L}_{char} = \sqrt{\|X_S - Y\|^2 + \epsilon^2}, \quad (38)$$

where ϵ is a small constant introduced for numerical stability, set to 10^{-3} in our implementation. Additionally, $\mathcal{L}_{edge}$ represents the edge-aware loss, defined as (eq.39):

$$\mathcal{L}_{edge} = \sqrt{\|\Delta(X_S) - \Delta(Y)\|^2 + \epsilon^2}, \quad (39)$$

where $\Delta(\cdot)$ denotes the Laplacian operator, which emphasizes high-frequency components such as edges in the image. This term encourages the preservation and accurate reconstruction of fine structural details in the restored images.

Quantitative results

Method	GoPro [53]				HIDE [69]			
	PSNR $\uparrow$		SSIM $\uparrow$		PSNR $\uparrow$		SSIM $\uparrow$	
Xu <i>et al.</i> [80]	21.00	(73.9%)	0.741	(84.2%)	-		-	
Hyun <i>et al.</i> [36]	23.64	(64.6%)	0.824	(76.7%)	-		-	
Whyte <i>et al.</i> [78]	24.60	(60.5%)	0.846	(73.4%)	-		-	
Gong <i>et al.</i> [27]	26.40	(51.4%)	0.863	(70.1%)	-		-	
DeblurGAN [42]	28.70	(36.6%)	0.858	(71.1%)	24.51	(52.4%)	0.871	(52.7%)
Nah <i>et al.</i> [53]	29.08	(33.8%)	0.914	(52.3%)	25.73	(45.2%)	0.874	(51.6%)
Zhang <i>et al.</i> [91]	29.19	(32.9%)	0.931	(40.6%)	-		-	
DeblurGAN-v2 [43]	29.55	(30.1%)	0.934	(37.9%)	26.61	(39.4%)	0.875	(51.2%)
SRN [71]	30.26	(24.1%)	0.934	(37.9%)	28.36	(25.9%)	0.915	(28.2%)
Shen <i>et al.</i> [69]	-		-		28.89	(21.2%)	0.930	(12.9%)
Gao <i>et al.</i> [25]	30.90	(18.3%)	0.935	(36.9%)	29.11	(19.2%)	0.913	(29.9%)
DBGAN [92]	31.10	(16.4%)	0.942	(29.3%)	28.94	(20.8%)	0.915	(28.2%)
MT-RNN [58]	31.15	(16.0%)	0.945	(25.5%)	29.15	(18.8%)	0.918	(25.6%)
DMPHN [88]	31.20	(15.5%)	0.940	(31.7%)	29.09	(19.4%)	0.924	(19.7%)
Suin <i>et al.</i> [70]	<u>31.85</u>	(8.9%)	<u>0.948</u>	(21.2%)	<u>29.98</u>	(10.7%)	<u>0.930</u>	(12.9%)
MPRNet (Ours)	32.66	(0.0%)	0.959	(0.0%)	30.96	(0.0%)	0.939	(0.0%)

Figure 51: Quantitative results on GoPro and HIDE datasets.

Method	RealBlur-R				RealBlur-J			
	PSNR $\uparrow$		SSIM $\uparrow$		PSNR $\uparrow$		SSIM $\uparrow$	
Hu <i>et al.</i> [33]	33.67	(23.4%)	0.916	(42.9%)	26.41	(23.2%)	0.803	(35.5%)
Nah <i>et al.</i> [53]	32.51	(33.0%)	0.841	(69.8%)	27.87	(9.1%)	0.827	(26.6%)
DeblurGAN [42]	33.79	(22.4%)	0.903	(50.5%)	27.97	(8.1%)	0.834	(23.5%)
Pan <i>et al.</i> [56]	34.01	(20.4%)	0.916	(42.9%)	27.22	(15.7%)	0.790	(39.5%)
Xu <i>et al.</i> [80]	34.46	(16.2%)	0.937	(23.8%)	27.14	(16.4%)	0.830	(25.3%)
DeblurGAN-v2 [43]	35.26	(8.1%)	0.944	(14.3%)	<u>28.70</u>	(0.0%)	0.866	(5.2%)
Zhang <i>et al.</i> [91]	35.48	(5.7%)	0.947	(9.4%)	27.80	(9.8%)	0.847	(17.0%)
SRN [71]	35.66	(3.7%)	0.947	(9.4%)	28.56	(1.6%)	<u>0.867</u>	(4.5%)
DMPHN [88]	<u>35.70</u>	(3.3%)	<u>0.948</u>	(7.7%)	28.42	(3.2%)	0.860	(9.3%)
MPRNet (Ours)	35.99	(0.0%)	0.952	(0.0%)	28.70	(0.0%)	0.873	(0.0%)
[‡] DeblurGAN-v2 [43]	36.44	(28.1%)	0.935	(56.9%)	29.69	(21.2%)	0.870	(40.0%)
[‡] SRN [71]	<u>38.65</u>	(7.3%)	<u>0.965</u>	(20.0%)	<u>31.38</u>	(4.3%)	<u>0.909</u>	(14.3%)
[‡] MPRNet (Ours)	39.31	(0.0%)	0.972	(0.0%)	31.76	(0.0%)	0.922	(0.0%)

Figure 52: Quantitative results on RealBlur-R and RealBlur-J datasets.

Qualitative results



Figure 53: Qualitative results on the GoPro dataset.

3.2.2 HINet: Half Instance Normalization Network for Image Restoration [2]

This paper primarily adopts a multi-stage architecture based on a CNN U-Net backbone (fig.54), and proposes an effective integration of normalization into convolutional blocks via the Half Instance Normalization (HIN) block to address low-level vision tasks such as image deblurring. While a variety of normalization techniques have been widely applied to high-level vision tasks—such as Batch Normalization (BN) and Instance-Batch Normalization (IBN) for image classification, Layer Normalization for DETR, and Group Normalization for FCOS—these methods may not be well-suited for low-level tasks.

In particular, Batch Normalization often suffers from instability when the batch size is small, which is a common setting for low-level vision problems due to memory constraints. To mitigate this issue, the authors incorporate Instance Normalization within the residual blocks, which is more robust in such scenarios.

Furthermore, the proposed network incorporates several modules from MPRNet, such as the Supervised Attention Module (SAM) and the Cross-Stage Feature Fusion (CSFF) mechanism. SAM facilitates the generation of stage-wise attention maps under supervision, which effectively guide the learning in subsequent stages. Meanwhile, CSFF enables information flow across stages, promoting better feature reuse and refinement throughout the network.

There are two contributions of this paper are summarized as follows:

- Proposed a novel residual block design, termed the HIN block, which effectively integrates Instance Normalization into the convolutional block. This design is specifically tailored for low-level vision tasks, addressing the limitations of traditional normalization methods—such as Batch Normalization—in scenarios with small batch sizes.
- Building upon the HIN block, the authors develop a multi-stage network, named HINet, for image restoration. The architecture incorporates key modules such as the Supervised Attention Module (SAM) and Cross-Stage Feature Fusion (CSFF) to progressively refine image quality across stages, demonstrating state-of-the-art performance on challenging image deblurring benchmarks.

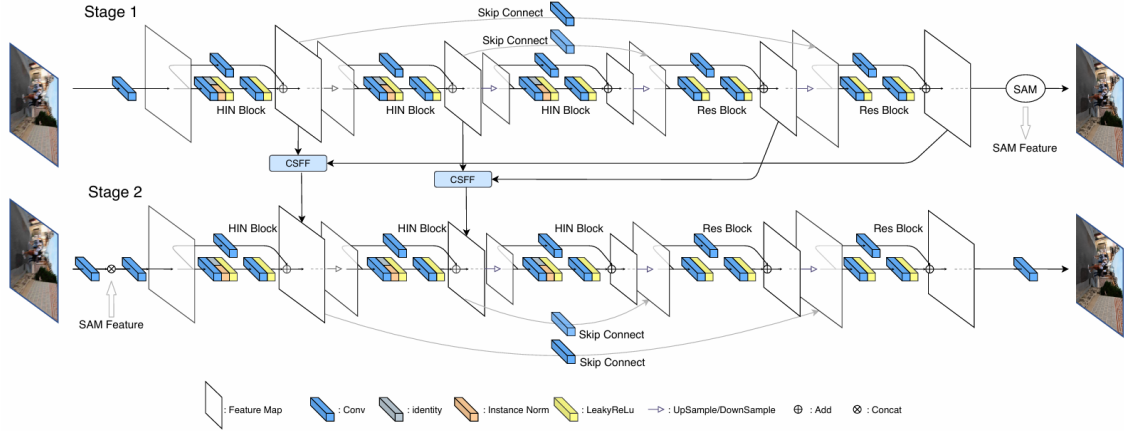


Figure 54: Proposed Half Instance Normalization Network (HINet). The encoder of each subnet-network contains Half Instance Normalization Blocks (HIN Block). For simplicity, we only show 3 layers of HIN Block in the figure, and HINet has a total of 5 layers. We adopt CSFF and SAM modules from MPRNet

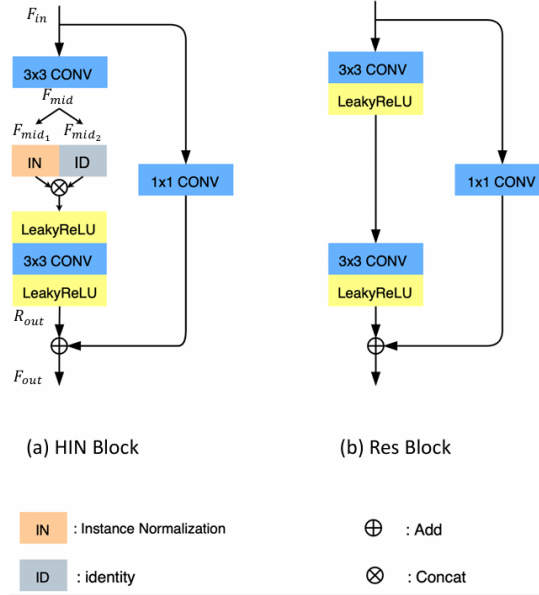


Figure 55: Proposed Half Instance Normalization Block (HIN Block) and ResBlock in details.

The authors first introduce the Half Instance Normalization block (HIN block), which serves as the fundamental building block throughout the proposed HINet architecture as illustrated in (fig.55). Within each HIN block, the intermediate feature map F_{mid} is split into two branches: (1) One half is processed through Instance Normalization (IN), which normalizes each channel independently by subtracting the mean and dividing by the standard deviation. (2) The other half retains its original, unnormalized form in order to preserve essential contextual information. The two branches are then concatenated and passed through a series of convolutional and ReLU layers to form the final output of the block. This design reflects the authors' insight that normalization improves training stability, while maintaining unnormalized features helps retain spatial and contextual integrity. Experimental results further suggest that the HIN block is particularly beneficial for capturing and enhancing shallow-level information.

Regarding the Supervised Attention Module (SAM), its design and function closely follow those in MPRNet and operate between stages as shown in (fig.50) SAM utilizes the ground truth to supervise the restoration result at each stage. The restored image is used to generate an attention

map that suppresses irrelevant or redundant information while emphasizing important features, thereby guiding more effective learning in the subsequent stage. The architecture also integrates Cross-Stage Feature Fusion (CSFF), again inspired by MPRNet, which is applied between consecutive stages as shown in (fig.49c).

The loss function used in this work is based on the sum of Peak Signal-to-Noise Ratio (PSNR) values computed at each stage of the network. Specifically, the objective is to maximize the PSNR of the intermediate outputs with respect to the ground truth. To enable gradient-based optimization, the negative PSNR is minimized, as expressed below (eq.40):

$$\mathcal{L} = - \sum_{i=1}^2 PSNR((R_i + X_i), Y), \quad (40)$$

where $R_i \in \mathbb{R}^{N \times C \times H \times W}$ denotes the final prediction (i.e., the residual) produced by subnetwork i , $X_i \in \mathbb{R}^{N \times C \times H \times W}$ is the input to subnetwork i , and $Y \in \mathbb{R}^{N \times C \times H \times W}$ is the ground truth image.

The addition of R_i and X_i yields the reconstructed image at stage i , and the PSNR is computed between this reconstruction and the ground truth. By summing across multiple stages, the network is encouraged to progressively improve the reconstruction quality during each stage of processing.

Quantitative results

Method	GoPro [31]	
	PSNR	SSIM
Xu et al. [48]	21.00	0.741
Hyun et al. [17]	23.64	0.824
Whyte et al. [46]	24.60	0.846
Gong et al. [13]	26.40	0.863
DeblurGAN [21]	28.70	0.858
Nah et al. [31]	29.08	0.914
Zhang et al. [62]	29.19	0.931
DeblurGAN-v2 [22]	29.55	0.934
SRN [39]	30.26	0.934
Gao et al. [12]	30.90	0.935
DBGAN [63]	31.10	0.942
MT-RNN [34]	31.15	0.945
DMPHN [57]	31.20	0.940
Suin et al. [37]	31.85	0.948
MPRNet [56]	<u>32.66</u>	0.959
HINet (ours)	32.71	0.959

Figure 56: Quantitative results on GoPro dataset.

Dataset	Method	PSNR	MACs(G)		Time(ms)	speedup
SIDD [1]	MPRNet [56]	39.71	573.50	100%	78.8	1×
	HINet 0.5×	39.82	42.88	7.5%	11.6	6.8×
	HINet	39.99	170.71	29.8%	27.0	2.9×
REDS [30]	MPRNet [56]	28.81	760.11	100%	90.1	1×
	HINet	28.79	170.71	22.5%	27.0	3.3×
GoPro [31]	MPRNet [56]	32.66	760.11	100%	90.1	1×
	HINet	32.71	170.71	22.5%	27.0	3.3×
Rain13k	MPRNet [56]	32.73	141.28	100%	37.4	1×
	HINet	33.03	170.71	120.8%	27.0	1.4×

Figure 57: Comparing the PSNR and MACs of MPRNet and ours. For Rain13k, we compare the average PSNR over Test100, Rain100H, Rain100L, Test2800 and Test1200. MACs and Time are estimated with the input size of $1 \times 3 \times 256 \times 256$. The proportion of the calculations and the speedup compared to MPRNet are also listed. Runtimes are computed with the Tesla V100 GPU

Qualitative results



Figure 58: Qualitative results

3.3 Self-attention

3.3.1 Restormer: Efficient Transformer for High-Resolution Image Restoration [19]

The authors identify two major limitations in conventional CNN-based architectures for image restoration tasks: (1) a limited receptive field, which restricts the model’s ability to capture long-range dependencies, and (2) the use of static weights during inference, which prevents the network from adapting dynamically to the content of each input. While self-attention mechanisms in Transformers offer a potential solution, their computational complexity grows quadratically with spatial resolution. To mitigate this, several prior works have adopted window-based or non-overlapping patch strategies. However, these approaches inherently limit the modeling of long-range pixel relationships, which are crucial for low-level vision tasks such as deblurring.

To address above issues, the authors propose to embed Transformer-based attention mechanisms into a U-Net-style architecture, introducing two key modules: (1) Multi-Dconv Head Transposed

Attention (MDTA). (2) Gated-Dconv Feed-Forward Network (GDFN). In the MDTA module, the authors improve upon standard self-attention by reshaping the query (Q) and key (K) representations before computing their matrix product, effectively reducing computational complexity while retaining contextual modeling ability. In the GDFN module enhances the standard feed-forward layer by incorporating a gating mechanism alongside GELU activation. This design enables adaptive control over feature flow, allowing the network to better regulate which information to retain or suppress during inference.

Furthermore, the overall architecture employs a progressive learning strategy, where the network gradually learns global representations from local features across stages. This hierarchical refinement improves both efficiency and performance in restoring high-frequency image details.

An overview of the architecture is illustrated in the following (fig.59).

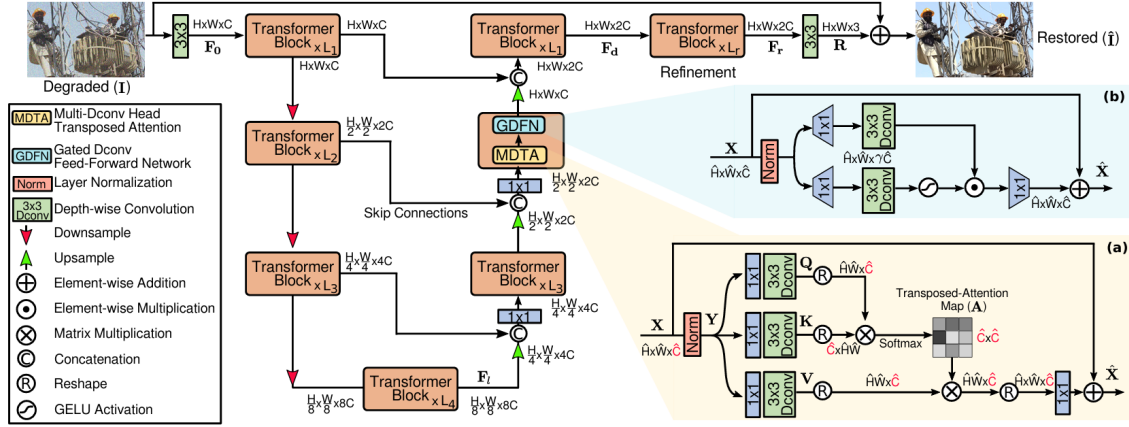


Figure 59: Architecture of Restormer for high-resolution image restoration. Our Restormer consists of multi scale hierarchical design in incorporating efficient Transformer blocks. The core modules of Transformer block are: (a) multi-Dconv head transposed attention (MDTA) that performs (spatially enriched) query-key feature interaction across channels rather than the spatial dimension, and (b) Gated-Dconv feed-forward network (GDFN) that performs controlled feature transformation, i.e., to allow useful information to propagate further.

The key contributions of this work are as follows:

- Proposed Restormer, an encoder-decoder Transformer architecture designed for image restoration tasks. Unlike previous Transformer-based models that rely on local windows or non-overlapping patches, Restormer is capable of learning multi-scale local-global representations without resorting to window partitioning, enabling efficient and effective modeling of both local details and global context.
- A novel attention mechanism, Multi-Dconv Head Transposed Attention (MDTA), is introduced to facilitate interaction between local and non-local pixel information. By restructuring the computation of self-attention through depthwise convolutions and transposed projections, MDTA significantly reduces complexity while preserving long-range dependencies.
- Proposed the Gated-Dconv Feed-Forward Network (GDFN) module is proposed to enable controlled feature transformation through a gating mechanism combined with non-linear activation (GELU). This enhances the model's capacity to regulate feature flow and adaptively emphasize informative regions during restoration.

The Multi-Dconv Head Transposed Attention (MDTA) module is designed to replace spatial attention with a more efficient channel-wise attention mechanism, thereby achieving linear computational complexity with respect to spatial dimensions. In addition, depthwise convolutions are incorporated to strengthen local contextual representations, enabling the model to balance both global dependency modeling and local detail preservation. As illustrated in (fig.59a), the input

feature map X is first normalized to obtain Y . Then, Y is passed through three parallel convolutional branches to generate the query (Q), key (K), and value (V) representations. Each branch consists of a 1×1 point-wise convolution followed by a 3×3 depthwise convolution, capturing local context efficiently. The resulting tensors are reshaped into: $Q, K \in \mathbb{R}^{\hat{H}\hat{W} \times \hat{C}}$, and $V \in \mathbb{R}^{\hat{C} \times \hat{H}\hat{W}}$

These are further transformed into $\hat{Q}$, $\hat{K}$, and $\hat{V}$, upon which the transposed attention map $A \in \mathbb{R}^{\hat{C} \times \hat{C}}$ is computed. This contrasts with traditional self-attention, which would compute a much larger attention map of size $\mathbb{R}^{\hat{H}\hat{W} \times \hat{H}\hat{W}}$. The overall MDTA process is described as follows (eq.41):

$$\begin{aligned} \hat{X} &= W_p \text{ Attention}(\hat{Q}, \hat{K}, \hat{V}) + X, \\ \text{Attention}(\hat{Q}, \hat{K}, \hat{V}) &= \hat{V} \cdot \text{Softmax}(\hat{K} \cdot \hat{Q} / \alpha), \end{aligned} \quad (41)$$

where X and $\hat{X}$ denote the input and output feature maps, respectively, and W_p is a learnable point-wise (1×1) convolution used to project the attended features. The scalar α is a learnable temperature parameter that stabilizes the dot-product operation before the softmax. Similar to standard multi-head self-attention, the feature channels are divided into multiple attention heads, allowing the model to learn diverse attention patterns in parallel.

The Feed-Forward Network (FFN) is a fundamental component in Transformer-based architectures. It operates independently and identically across all spatial positions, transforming the features through a sequence of point-wise operations and non-linear activations. Conventionally, FFNs expand the feature dimension by a factor γ (typically $\gamma = 4$), apply a non-linear activation such as GELU, and project the features back to the original channel dimension. The Gated-Dconv Feed-Forward Network (GDFN) enhances the standard FFN in two key ways: (1) Gating Mechanism: Introduces a non-linear control over information flow. It enables the model to modulate features dynamically through element-wise gating, enhancing expressive capacity and learning flexibility. (2) Depthwise Convolutions: Integrates depthwise convolutions to capture local spatial structure, which is especially beneficial for low-level image restoration tasks where spatial detail is critical. The overall computation within GDFN is illustrated in (fig.59b) and can be formulated as (eq.42):

$$\begin{aligned} \hat{X} &= W_p \text{ Gating}(X) + X, \\ \text{Gating}(X) &= \phi(W_d W_p (\text{LN}(X))) \odot W_d W_p (\text{LN}(X)), \end{aligned} \quad (42)$$

where X and $\hat{X}$ denote the input and output feature maps, respectively, LN is Layer Normalization, W_p and W_d are point-wise and depthwise convolution weights, ϕ is the GELU activation function, and $\odot$ represents element-wise multiplication (the gating operation). This design allows the feed-forward module to simultaneously model both channel-wise transformations and local spatial dependencies, leading to improved feature representation and restoration quality.

In addition, the authors adopt a progressive learning strategy during training to enhance both the efficiency and effectiveness of feature learning in Transformer-based image restoration. This approach involves dynamically adjusting the patch size used for training over the course of optimization. In the early stages of training, relatively small image patches are used. This encourages the model to focus on learning basic features and local priors, which are essential for establishing low-level visual understanding. As training progresses, the patch size is gradually increased, allowing the model to capture global context and fine-grained details more effectively. This strategy also enables the Transformer to better adapt to varying image resolutions, which is beneficial for real-world restoration scenarios. To maintain training efficiency and computational balance, the authors reduce the batch size correspondingly as the patch size increases. This ensures that the overall memory footprint and training time remain approximately constant, allowing the benefits of progressive learning to be achieved without compromising resource efficiency.

Loss function is using L_1 loss.

Quantitative results

Method	GoPro [56]		HIDE [67]		RealBlur-R [65]		RealBlur-J [65]	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Xu <i>et al.</i> [84]	21.00	0.741	-	-	34.46	0.937	27.14	0.830
DeblurGAN [38]	28.70	0.858	24.51	0.871	33.79	0.903	27.97	0.834
Nah <i>et al.</i> [56]	29.08	0.914	25.73	0.874	32.51	0.841	27.87	0.827
Zhang <i>et al.</i> [98]	29.19	0.931	-	-	35.48	0.947	27.80	0.847
DeblurGAN-v2 [39]	29.55	0.934	26.61	0.875	35.26	0.944	28.70	0.866
SRN [72]	30.26	0.934	28.36	0.915	35.66	0.947	28.56	0.867
Shen <i>et al.</i> [67]	-	-	28.89	0.930	-	-	-	-
Gao <i>et al.</i> [23]	30.90	0.935	29.11	0.913	-	-	-	-
DBGAN [100]	31.10	0.942	28.94	0.915	33.78	0.909	24.93	0.745
MT-RNN [58]	31.15	0.945	29.15	0.918	35.79	0.951	28.44	0.862
DMPHN [94]	31.20	0.940	29.09	0.924	35.70	0.948	28.42	0.860
Suin <i>et al.</i> [71]	31.85	0.948	29.98	0.930	-	-	-	-
SPAIR [61]	32.06	0.953	30.29	0.931	-	-	28.81	0.875
MIMO-UNet+ [14]	32.45	0.957	29.99	0.930	35.54	0.947	27.63	0.837
IPT [13]	32.52	-	-	-	-	-	-	-
MPRNet [93]	32.66	0.959	30.96	0.939	35.99	0.952	28.70	0.873
Restormer	32.92	0.961	31.22	0.942	36.19	0.957	28.96	0.879

Figure 60: Quantitative results.

Qualitative results



Figure 61: Qualitative results on GoPro dataset.

3.3.2 Efficient Frequency Domain-based Transformers for High-Quality Image Deblurring [6]

This paper explores the integration of self-attention mechanisms within U-Net-style architectures by introducing a novel frequency-domain formulation. Specifically, the authors leverage Fast Fourier Transform (FFT) to reformulate both the self-attention mechanism and the feed-forward network in the frequency domain, thereby improving computational efficiency and representational power.

The motivation stems from the observation that convolution in the spatial domain corresponds to element-wise multiplication in the frequency domain, while standard self-attention suffers from quadratic complexity due to large matrix multiplications. To address this, the authors propose the Frequency domain-based Self-Attention Solver (FSAS), which enables full-image attention

with reduced computational cost. This formulation eliminates the need for window-based attention, thereby avoiding the locality constraints and fragmentation of context that such approaches typically introduce. In addition, the authors propose the Discriminative Frequency-domain Feed-Forward Network (DFFN). This module enhances the traditional feed-forward network by applying FFT-based operations and introducing a learnable quantization matrix to preserve and selectively emphasize important frequency components. The design of DFFN is inspired by the JPEG compression pipeline, where frequency information is quantized to emphasize visually significant features.

The overall network adopts an asymmetric encoder-decoder architecture within the U-Net framework. In the encoder, only the DFFN module is employed, focusing on efficient frequency-domain feature transformation. In contrast, the decoder utilizes both FSAS and DFFN, allowing it to reconstruct high-quality images by leveraging full-range attention and frequency-aware transformations.

The key contributions of this work are summarized as follows:

- A novel formulation of self-attention Frequency Domain-based Self-Attention Solver (FSAS) in the frequency domain is proposed, which enables full-image attention with significantly reduced computational complexity. By leveraging the properties of Fourier transforms, FSAS avoids the limitations of window-based attention while preserving global contextual modeling.
- Discriminative Frequency-domain Feed-Forward Network (DFFN) is proposed. It is inspired by the JPEG compression pipeline, the DFFN module introduces a learnable quantization matrix in the frequency domain. This design enables selective enhancement of important frequency components, leading to improved discriminative capability and detail preservation in the restored images.
- The network adopts a non-symmetric U-Net structure, where the encoder utilizes only the DFFN module for efficient feature encoding, while the decoder integrates both FSAS and DFFN to enhance reconstruction quality through frequency-aware attention and transformation mechanisms.

Below (fig.62) is the proposed model.

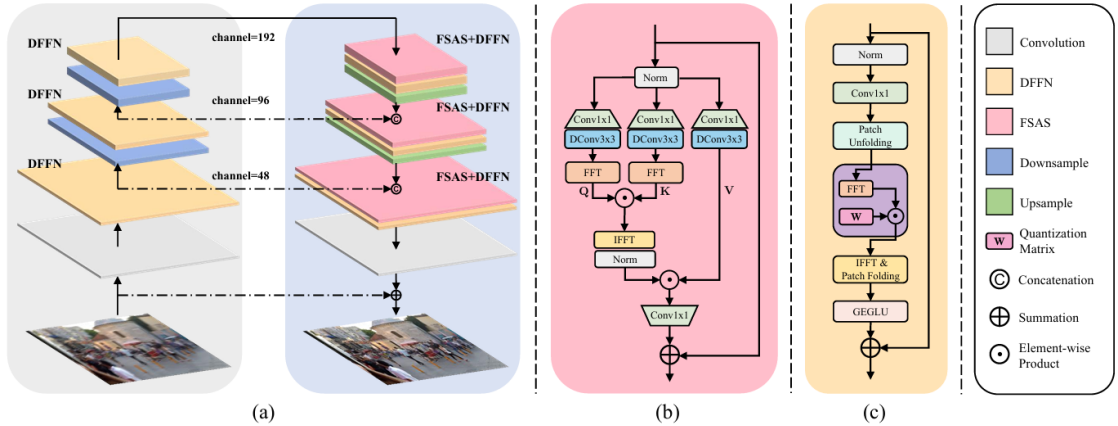


Figure 62: Network architectures. (a) The proposed asymmetric encoder-decoder network that only contains DFFN in the encoder module and both FSAS and DFFN in the decoder module for image deblurring. (b) The proposed FSAS module. (c) The proposed DFFN module.

In conventional Vision Transformer architectures, self-attention is computed over flattened image patches derived from input feature maps. Given an input feature tensor $X \in \mathbb{R}^{H \times W \times C}$ with spatial resolution $H \times W$ and C channels, the feature map is first projected into query, key, and value representations through learned linear transformations (eq.43):

$$F_q = W_q X, F_k = W_k X, F_v = W_v X, \quad (43)$$

where W_q , W_k , and W_v are learned projection matrices. These feature maps are then unfolded into a set of image patches $q_i i = 1^N$, $k_i i = 1^N$, and $v_i i = 1^N$, where N is the total number of extracted patches. The patches are reshaped to construct the query, key, and value matrices (eq.44):

$$Q = \mathcal{R}(\{q_i\}_{i=1}^N), \quad K = \mathcal{R}(\{k_i\}_{i=1}^N), \quad V = \mathcal{R}(\{v_i\}_{i=1}^N), \quad (44)$$

where $\mathcal{R}$ denotes a reshape operation that converts each patch into a vector of dimension CH_pW_p , with H_p and W_p being the height and width of each patch. This yields $Q, K, V \in \mathbb{R}^{N \times (CH_pW_p)}$. The standard self-attention is then computed as (eq.45):

$$V_{att} = \text{softmax}\left(\frac{QK^T}{\sqrt{CH_pW_p}}\right)V, \quad (45)$$

where QK^T represents a matrix multiplication that incurs quadratic time and space complexity, specifically $O(N^2)$ for memory and $O(N^2C)$ for computation. This cost becomes significant for high-resolution images.

Frequency Domain-based Self-Attention Solver (FSAS) is motivated by the observation that the matrix multiplication QK^T is structurally similar to convolution, the authors propose an alternative formulation in the frequency domain using the Fast Fourier Transform (FFT). This reformulation reduces the complexity while preserving full-image attention. In FSAS, the attention map A is computed via element-wise operations in the frequency domain (eq.46):

$$A = \mathcal{F}^{-1}(\mathcal{F}(Q)\overline{\mathcal{F}(K)}), \quad (46)$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote the forward and inverse FFT, respectively, and $\overline{(\cdot)}$ is the complex conjugate. This operation efficiently captures global correlations analogous to spatial-domain attention. The final attended features are obtained as (eq.47):

$$V_{att} = \mathcal{L}(A)V_v, \quad (47)$$

where $\mathcal{L}(\cdot)$ denotes a Layer Normalization operation applied to the attention map A before it is used to modulate the value features V_v . This frequency-domain approach not only maintains global receptive fields but also significantly reduces computational burden, making it especially advantageous for high-resolution image restoration tasks. The final output of the Frequency domain-based Self-Attention Solver (FSAS) is computed as (eq.48):

$$X_{att} = X + \text{Conv}_{1 \times 1}(V_{att}), \quad (48)$$

where $\text{Conv}_{1 \times 1}(\cdot)$ denotes a 1×1 point-wise convolution, applied to project the attended features V_{att} back to the original channel dimension, followed by residual addition with the input X . By computing attention in the frequency domain, FSAS significantly reduces the computational complexity. Specifically, the space and time complexity are reduced from $\mathcal{O}(N^2)$ and $\mathcal{O}(N^2C)$ in conventional attention to: Space complexity: $\mathcal{O}(N)$, Time complexity: $\mathcal{O}(NC \log N)$. This makes FSAS highly scalable for high-resolution image processing tasks.

In the feed-forward component, the authors propose the Discriminative Frequency-domain Feed-Forward Network (DFFN), inspired by the quantization process in JPEG compression. As illustrated in (fig.62c), a learnable frequency-weighting matrix W is introduced. The feature map is transformed into the frequency domain using FFT, modulated via element-wise multiplication with W , and then transformed back. This operation enables the model to preserve and emphasize informative frequency components, enhancing the discriminative capacity of the feed-forward module.

Empirical observations revealed that features extracted in shallower layers tend to contain blur artifacts, which can negatively impact the performance of the deblurring task. Consequently, the authors adopt an asymmetric encoder-decoder architecture: In the encoder, only DFFN is used, excluding FSAS, to avoid reinforcing blur-influenced attention. In the decoder, both FSAS and DFFN are applied, allowing the network to leverage global attention and frequency-aware transformation at deeper stages, where high-level semantic and contextual cues are more prominent and reliable. This design balances computational efficiency and representational robustness, leading to improved restoration performance.

Methods	FSAS in enc&dec	FSAS in dec (Ours)
PSNRs	33.56	33.73
SSIMs	0.9653	0.9663

Figure 63: Quantitative evaluations of the asymmetric encoder decoder network on the GoPro dataset.

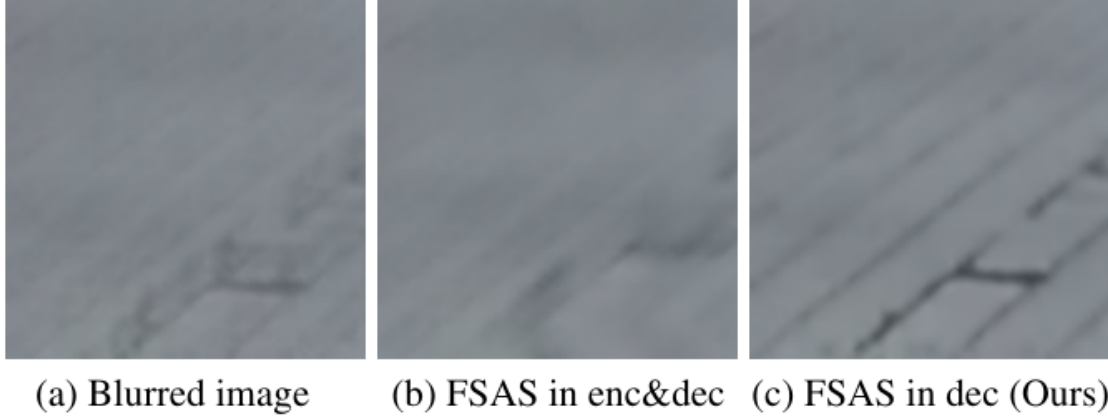


Figure 64: Effectiveness of the asymmetric encoder-decoder network on image deblurring.

The loss function used in this work is identical to that of MIMO-UNet. This choice ensures consistency in training objectives and enables fair comparison between architectures. Please refer to the original MIMO-UNet paper for detailed formulation and implementation specifics.

Quantitative results

Methods	PSNRs	SSIMs	Parameters (M)	Avg. runtime
DeblurGAN-v2 [12]	29.55	0.9340	60.9	0.04s
SRN [22]	30.26	0.9342	6.8	0.07s
DMPHN [32]	31.20	0.9453	21.7	0.21s
SAPHN [21]	31.85	0.9480	23.0	-
MIMO-Unet+ [4]	32.45	0.9567	16.1	0.02s
MPRNet [31]	32.66	0.9589	20.1	0.09s
IPT [2]	-	-	114	0.50s
DeepRFT+ [15]	33.23	0.9632	23.0	0.09s
Restormer [30]	32.92	0.9611	26.1	0.08s
Uformer-B [27]	33.06	0.9670	50.9	0.07s
Stripformer [24]	33.08	0.9624	19.7	0.04s
MPRNet-local [5]	33.31	0.9637	20.1	0.11s
Restormer-local [5]	33.57	0.9656	26.1	0.42s
NAFNet [3]	33.71	0.9668	67.9	0.04s
Ours	34.21	0.9692	16.6	0.13s

Figure 65: Quantitative evaluations on the GoPro dataset. The average runtime is tested on images with size of 256×256 pixels.

Qualitative results

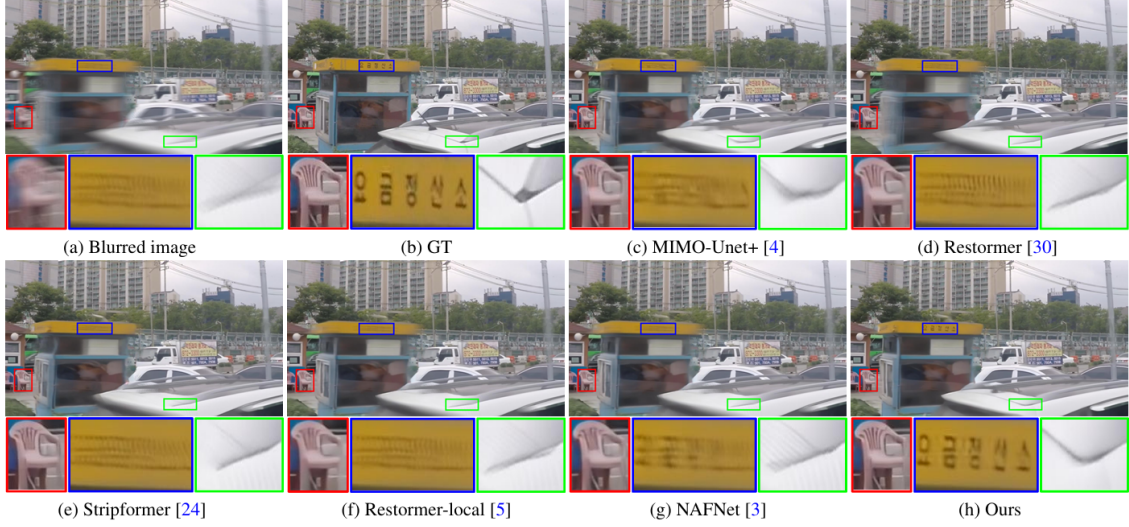


Figure 66: Quantitative results on the GoPro dataset.

3.4 MLP

3.4.1 MAXIM:Multi-Axis MLP for Image Processing [16]

This paper proposes a multi-stage image restoration framework based on the MIMO-UNet architecture, where the core computational blocks are redesigned using MLP-based modules instead of traditional CNNs or self-attention mechanisms. The goal is to address the inherent limitations of these conventional designs in the context of low-level vision tasks. While Vision Transformers (ViT) and MLP-Mixers have demonstrated impressive performance on high-level tasks such as image classification, their direct application to low-level tasks (e.g., image restoration) presents several challenges: Inability to handle high-resolution images effectively due to patch-based input representation, which often introduces patch boundary artifacts. Limited attention capability: many transformer variants, such as Swin Transformer, restrict attention to local windows, resulting in a lack of global contextual modeling. CNN-based models, while effective in extracting local features, inherently lack the ability to perform global feature mixing due to their limited receptive fields.

To overcome above limitations, the authors propose the Multi-axis Gated MLP, a novel building block that leverages multiple computation branches along different spatial axes to capture both local and global dependencies. The use of fully convolutional MLPs eliminates the need for patch tokenization and supports input images of arbitrary resolution, thereby mitigating the patch artifact issue. Additionally, the authors introduce the Cross-Gating Block, which also incorporates the Multi-axis Gated MLP to perform cross-feature fusion across different stages or branches, serving as an alternative to conventional cross-attention mechanisms.

The overall network follows a multi-stage hierarchical design, where features are progressively refined across stages with intermediate supervision. The architecture maintains the strength of MIMO-UNet in multi-scale restoration while benefiting from the enhanced global-local feature modeling capabilities of gated MLPs. An overview of the architecture is illustrated in the following (fig.67).

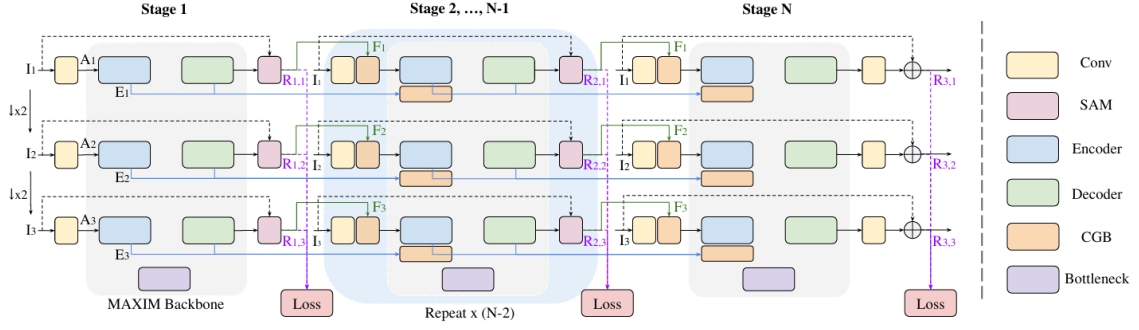


Figure 67: We adopt a general multi-stage framework to improve the performance of MAXIM for challenging restoration tasks. We employ the supervised attention module (SAM) and cross-stage feature fusion to help later stages learn. Unlike previous approaches, our MAXIM backbone attains global perception at each layer in each stage due to the proposed multi-axis MLP approaches, making it more powerful in learning global interactions in both low-level and high-level features.

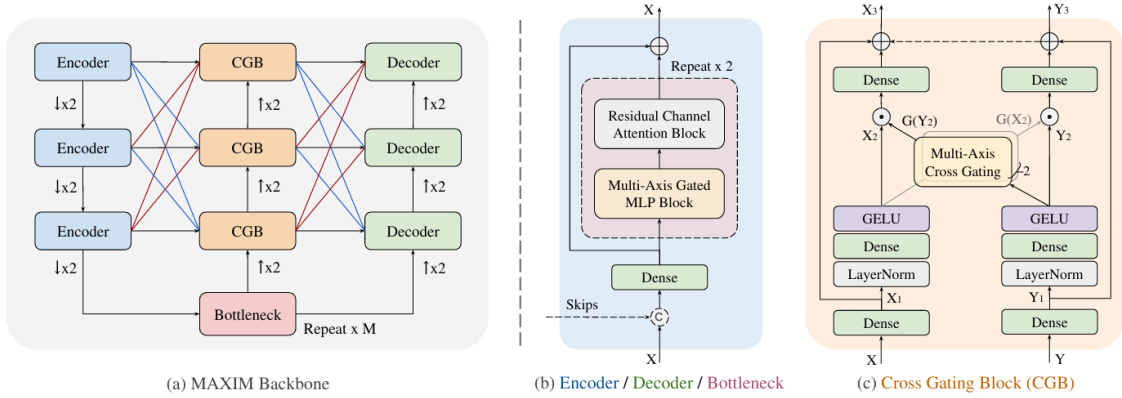


Figure 68: MAXIM architecture. We take (a) an encoder-decoder backbone with each (b) encoder, decoder, and bottleneck containing a multi-axis gated MLP block (Fig.3) as well as a residual channel attention block. The model is further boosted by (c) across gating block which allows global contextual features to gate the skip connections.

The main contributions of this work are summarized as follows:

- **MAXIM: Multi-Axis MLP for Image Processing** The authors propose MAXIM, a novel MLP-based building block for image restoration, which is embedded into a multi-scale and multi-stage architecture. Unlike traditional CNNs or Transformers, MAXIM adopts pure MLP-based computation to handle low-level vision tasks effectively, leveraging the MLP’s flexibility in processing images of arbitrary resolution.
- **Multi-Axis Gated MLP with Linear Complexity** A key innovation is the Multi-Axis Gated MLP, which captures both global and local dependencies via separate computational branches along different spatial axes. This design achieves linear computational complexity with respect to image size, addressing the scalability issues of standard self-attention while preserving global contextual awareness.
- **Cross-Gating Block for Feature Fusion** The authors introduce a Cross-Gating Block, designed to decouple and fuse features from different branches or stages. This module also leverages the Multi-Axis Gated MLP, enabling efficient and flexible cross-feature interaction without relying on conventional cross-attention mechanisms.

The Multi-Axis Gated MLP (gMLP) Block is the core component of the proposed MAXIM architecture. Its main objective is to effectively capture both local and global dependencies by processing features along multiple spatial axes through separate computational branches. As illustrated in (fig.69), the block is composed of two parallel branches: a Local Branch and a Global Branch.

Local Branch: In this branch, the input feature map is divided into relatively small non-overlapping blocks. Each block undergoes a gated MLP (gMLP) operation, which enables spatial mixing within localized regions. After processing, the blocks are recombined (folded) back into the original spatial arrangement.

Global Branch: Here, the input is divided into larger grids, enabling the capture of long-range dependencies across the image. Similar to the local branch, each grid is processed independently using a gMLP, and the resulting features are recombined to match the original spatial layout.

Finally, the outputs from the local and global branches are concatenated along the channel dimension, and a dense (fully connected) layer is applied to fuse the information from both branches.

This multi-axis design allows the model to simultaneously leverage fine-grained local structures and broad contextual information, all while maintaining linear computational complexity with respect to image size.

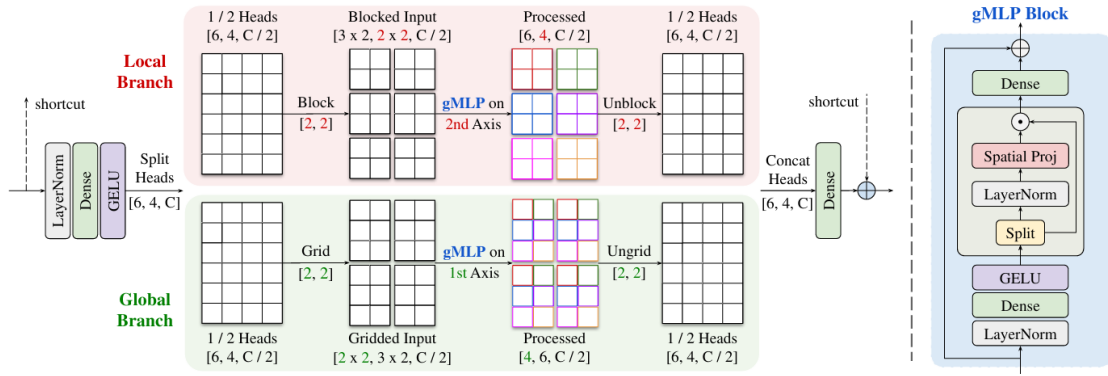


Figure 69: Multi-axis gated MLP block (best viewed in color). The input is first projected to a $[6, 4, C]$ feature, then split into two heads. In the local branch, the half head is blocked into $3 \times 2 \times 2$ non-overlapping $[2, 2, C/2]$ patches, while we grid the other half using a 2×2 grid in the global branch. We only apply the gMLP block (illustrated in the right gMLP Block) on a single axis of each branch- the 2nd axis for the local branch and the 1st axis for the global branch, while shared along the other spatial dimensions. The gMLP operators, which run in parallel, correspond to local and global (dilated) attended regions, as illustrated with different colors (i.e., the same color are spatially mixed using the gMLP operator). Our proposed block expresses both global and local receptive fields on arbitrary input resolutions.

The Cross Gating MLP Block is designed to facilitate information exchange and integration across different stages or spatial scales in the network. It serves as a lightweight and effective alternative to cross-attention mechanisms, making use of multi-axis gated MLPs to selectively control feature interaction between two input streams. The overall structure is illustrated in (fig.69c). The following is the pipeline.

Let X and Y be two input feature tensors from different stages or scales. The cross gating process is as follows:

1. Initial Feature Transformation:

Each input is first passed through a dense layer:

$$X_1 = \text{Dense}(X), \quad Y_1 = \text{Dense}(Y), \quad (49)$$

where $\text{Dense}(\cdot)$ denotes a point-wise fully connected layer.

2. Nonlinear Activation and Layer Normalization:

The features are normalized and activated via GELU:

$$X_2 = \sigma(W_1 \text{LN}(X_1)), \quad Y_2 = \sigma(W_2 \text{LN}(Y_1)), \quad (50)$$

where $\text{LN}(\cdot)$ is Layer Normalization, $\sigma(\cdot)$ denotes the GELU activation, and W_1, W_2 are learnable projection matrices.

3. Cross Gating Mechanism:

Cross gating is applied by modulating each branch using gating weights computed from the other branch:

$$\hat{X} = X_2 \odot G(Y_2), \quad \hat{Y} = Y_2 \odot G(X_2), \quad (51)$$

where $\odot$ denotes element-wise multiplication, and $G(\cdot)$ is the gating function that extracts attention weights across multiple axes. The gating function $G(x)$ is defined as:

$$\begin{aligned} [z_1, z_2] &= \sigma(W_6 \text{LN}(x)) \\ G(x) &= W_5([W_3 \text{Block}_b(z_1), W_4 \text{Grid}_d(z_2)]), \end{aligned} \quad (52)$$

where z_1 and z_2 are channel-wise splits of z , $[\cdot, \cdot, \cdot, \cdot]$ denotes channel-wise concatenation, and $\text{Block}_b(\cdot)$ and $\text{Grid}_d(\cdot)$ represent local and global spatial projections, respectively. The projection matrices W_3, W_4 , and W_5 are learned during training.

4. Output with Residual Connection:

The gated outputs are passed through additional dense layers and combined with the original features via residual addition:

$$X_3 = X_1 + W_7 \hat{X}, \quad Y_3 = Y_1 + W_8 \hat{Y}, \quad (53)$$

where W_7 and W_8 are projection matrices to match dimensions.

The network is supervised using a combination of Charbonnier loss and Frequency Reconstruction loss, applied across all stages and scales of the model outputs. The total loss is formulated as (eq:54):

$$\mathcal{L} = \sum_{s=1}^S \sum_{n=1}^N [\mathcal{L}_{char}(R_{s,n}, T_n) + \lambda \mathcal{L}_{freq}(R_{s,n}, T_n)], \quad (54)$$

where S is stage, N is scale, R is deblurring images, T is ground truth, $\mathcal{L}_{char}$ is Charbonnier loss, and $\mathcal{L}_{freq}$ is frequency reconstruction loss. The Charbonnier loss is a smooth variant of the L1 loss, which is more robust to outliers. It is defined as (eq:55):

$$\mathcal{L}_{char}(R, T) = \sqrt{\|R - T\|^2 + \epsilon^2}, \quad (55)$$

where ϵ is a small constant to ensure numerical stability. To encourage the preservation of high-frequency details, the Frequency Reconstruction loss compares the magnitude spectra of the restored and ground-truth images in the Fourier domain (eq:56):

$$\mathcal{L}_{freq}(R, T) = \|\mathcal{F}(R) - \mathcal{F}(T)\|_1, \quad (56)$$

where $\mathcal{F}(\cdot)$ denotes the Fourier transform, and $\|\cdot\|$ is the L_1 norm applied to the frequency domain representations. This dual-loss formulation helps the model produce deblurred outputs that are both perceptually smooth and rich in fine structural details.

The author conduct a simplified analysis to observe how varying the number of stages, channel width, and network depth affects model parameters, FLOPs, and PSNR on (fig.70). Using a standard U-Net as the baseline.

	S	W	D	PSNR	Params	FLOPs
Base	1	32	3	31.08	6.1M	93.6G
Wider	1	64	3	32.09	19.4M	309.9G
	1	96	3	32.31	41.7M	648.9G
Deeper	1	32	4	31.17	19.8M	121.6G
	1	32	5	31.43	75.0M	153.4G
More	2	32	3	31.82	14.1M	216.4G
stages	3	32	3	32.44	22.2M	339.2G

Figure 70: Ablation studies. Why multi-stage? S: stages, W: width (channel), and D: deep (multi-scale number)

Quantitative results

Method	GoPro [62]		HIDE [81]		Average	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
DeblurGAN [40]	28.70	0.858	24.51	0.871	26.61	0.865
Nah <i>et al.</i> [62]	29.08	0.914	25.73	0.874	27.41	0.894
Zhang <i>et al.</i> [118]	29.19	0.931	-	-	-	-
DeblurGAN-v2 [41]	29.55	0.934	26.61	0.875	28.08	0.905
SRN [86]	30.26	0.934	28.36	0.915	29.31	0.925
Shen <i>et al.</i> [81]	-	-	28.89	0.930	-	-
Gao <i>et al.</i> [28]	30.90	0.935	29.11	0.913	30.01	0.924
DBGAN [119]	31.10	0.942	28.94	0.915	30.02	0.929
MT-RNN [69]	31.15	0.945	29.15	0.918	30.15	0.932
DMPHN [113]	31.20	0.940	29.09	0.924	30.15	0.932
Suin <i>et al.</i> [83]	31.85	0.948	29.98	0.930	30.92	0.939
MPRNet [111]	32.66	<u>0.959</u>	<u>30.96</u>	<u>0.939</u>	<u>31.81</u>	<u>0.949</u>
Pretrained-IPT [15]	32.58	-	-	-	-	-
MIMO-UNet+ [20]	32.45	0.957	29.99	0.930	31.22	0.944
HINet [16]	<u>32.71</u>	<u>0.959</u>	30.32	0.932	31.52	0.946
MAXIM-3S	32.86	0.961	32.83	0.956	32.85	0.959

Figure 71: Quantitative results on the GoPro and HIDE datasets. MAXIM-"X"S: X is represent the number of stages.

Method	RealBlur-R [79]		RealBlur-J [79]		Average	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Hu <i>et al.</i> [33]	33.67	0.916	26.41	0.803	30.04	0.860
Nah <i>et al.</i> [62]	32.51	0.841	27.87	0.827	30.19	0.834
DeblurGAN [40]	33.79	0.903	27.97	0.834	30.88	0.869
Pan <i>et al.</i> [68]	34.01	0.916	27.22	0.790	30.62	0.853
Xu <i>et al.</i> [103]	34.46	0.937	27.14	0.830	30.8	0.884
DeblurGAN-v2 [41]	35.26	0.944	28.70	0.866	31.98	0.905
Zhang <i>et al.</i> [118]	35.48	0.947	27.80	0.847	31.64	0.897
SRN [86]	35.66	0.947	28.56	0.867	32.11	0.907
DMPHN [113]	35.70	0.948	28.42	0.860	32.06	0.904
MPRNet [111]	35.99	0.952	28.70	0.873	32.35	0.913
MAXIM-3S	35.78	0.947	28.83	0.875	32.31	0.911
[†] DeblurGAN-v2	36.44	0.935	29.69	0.870	33.07	0.903
[†] SRN [86]	38.65	0.965	31.38	0.909	35.02	0.937
[†] MPRNet [111]	39.31	0.972	31.76	0.922	35.54	0.947
[†] MIMO-UNet+ [20]	-	-	32.05	0.921	-	-
[†] MAXIM-3S	39.45	0.962	32.84	0.935	36.15	0.949

Figure 72: Quantitative results on the RealBlur datasets.

Qualitative results

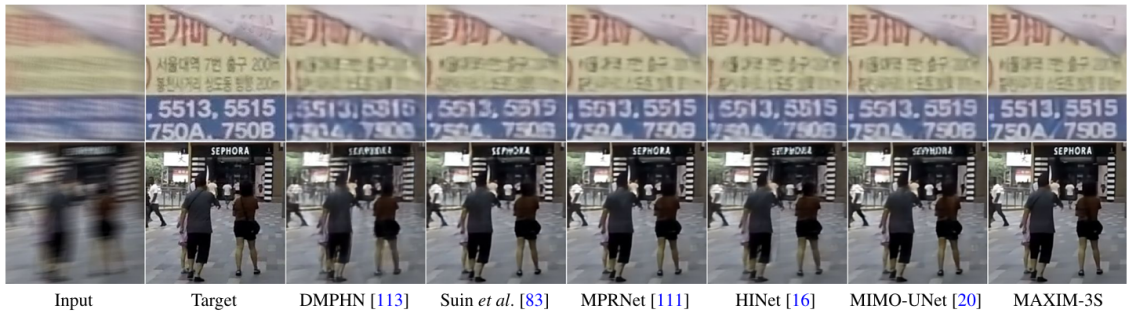


Figure 73: Qualitative results on the GoPro dataset.

3.5 SSM

3.5.1 Efficient Visual State Space Model for Image Deblurring [7]

This paper introduces a novel deblurring network that integrates a State Space Model (SSM) into the U-Net architecture to address key limitations of both CNNs and Transformers.

CNNs often struggle to capture spatially varying patterns and global dependencies, while Transformer-based models suffer from quadratic time and space complexity due to self-attention. Although variants such as window-based, transposed, and FFT-based attention reduce complexity, they often compromise on capturing non-local and spatial information. Recent developments

in SSMs offer a promising alternative by enabling linear or near-linear complexity while approximating attention-like behavior. However, most existing SSMs are tailored for 1D data. When extended to 2D, methods either: Flatten the 2D data, losing spatial structure and local context, or Use multi-directional scanning, which increases computational cost.

To overcome these challenges, the authors propose a simple geometric transformation-based scanning strategy that allows the SSM to operate efficiently on 2D feature maps. This design preserves spatial structure and enhances the ability to model non-local dependencies, without introducing significant computational overhead.

The overall deblurring process is formulated as (eq.57):

$$I_{deblur} = R + I_{blur} = \mathcal{N}(I_{blur}) + I_{blur} \quad (57)$$

where $\mathcal{N}(\cdot)$ denotes encoder-decoder network, Up- and Down-sampling: bilinear interpolation, and Skip connection add 1×1 conv. The following (fig.74) is the overall model.

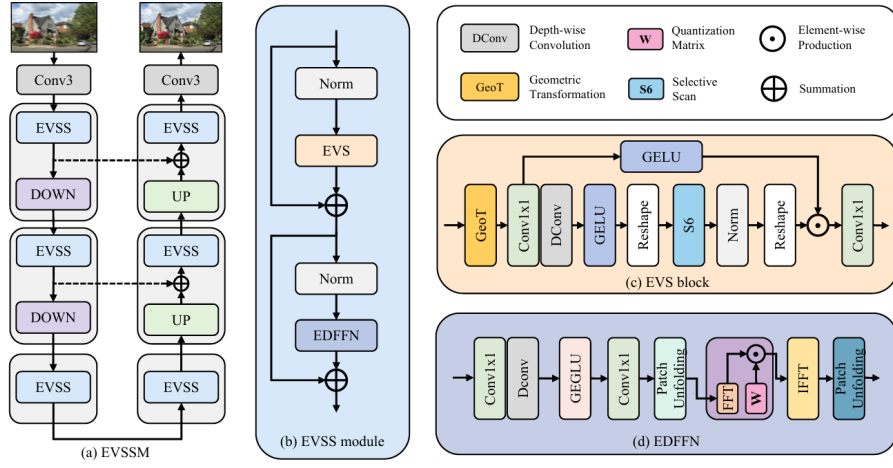


Figure 74: Efficient visual state space model. To efficiently restore high-quality images with SSMs, we propose an effective EVSS module that involves an efficient EVS block and an efficient EDFFN block. Ageometric transformation is employed at the beginning of each EVS block to facilitate more useful information exploration in the following selective scan with a minimal increase in computational costs.

The key contributions of this work are as follows:

- The authors propose a novel state space model tailored for visual tasks, which effectively captures both local and non-local spatial dependencies with linear complexity.
- Efficient 2D Scanning Strategy. A simple yet efficient scanning mechanism based on geometric transformations is introduced, enabling the state space model to operate on 2D feature maps without flattening or expensive multi-directional operations.

The Efficient Visual Scan (EVS) Block, as illustrated in (fig.74c), is designed to enhance the ability of the state space model to capture non-local spatial dependencies with minimal computational overhead. The core component of this block is the Geometric Transformation (GeoT) module, which leverages simple operations such as flipping and transposition to simulate multi-directional scanning without significantly increasing runtime complexity. The transformation applied at each EVS block is defined as (eq.58):

$$G = \begin{cases} Transponse(F_{in}) & \text{if } i\%2 = 0, \\ Flip(F_{in}) & \text{if } i\%2 = 1. \end{cases} \quad (58)$$

where i is the index of the current EVS block in the overall architecture. These geometric transformations do not alter the translation-invariant property of convolutions, but instead modify the

input orientation for the state space model’s selective scanning mechanism (S6). This prevents the SSM from being restricted by its recursive structure and enhances its capability to model non-local spatial correlations. Notably, after every four EVS blocks, the transformed input naturally returns to its original orientation, ensuring consistency.

The Efficient Discriminative Frequency FFN (EDFFN) is inspired by the DFFN module from FFTformer, which learns to retain important frequency components through a learnable quantization mechanism that is shown on (fig.74d). However, unlike DFFN, which is integrated within the middle of the FFN block, the EDFFN module is placed at the end of the FFN to act as a frequency-domain filter. This design allows EDFFN to operate as a final gate, selectively preserving or suppressing frequency-domain features based on their contribution to image restoration. By focusing frequency selection post-FFN, EDFFN improves the discriminative capacity of the network while maintaining computational efficiency.

The training objective follows the loss formulation used in MIMO-UNet, which combines both content loss and frequency reconstruction loss to ensure spatial fidelity and frequency consistency in the deblurred outputs. The total loss is defined as (eq.59):

$$\mathcal{L} = \|I_{deblur} - I_{gt}\|_1 + \lambda \|\mathcal{F}(I_{deblur}) - \mathcal{F}(I_{gt})\|_1, \quad (59)$$

where I_{deblur} and I_{gt} denotes the deblurring image and ground image, λ is hyperparameter, $\mathcal{F}(\cdot)$ denotes DFT

Quantitative results

Method	SRN [40]	DMPHN [55]	MIMO-UNet+ [5]	MPRNet [53]	MAXIM [43]	Uformer [48]	Restormer [54]
PSNR (dB)	30.26	31.20	32.45	32.66	32.86	33.06	32.92
SSIM	0.9342	0.9453	0.9567	0.9589	0.9616	0.9670	0.9611
Method	Stripformer [42]	Restormer-local [6]	NAFNet [3]	GRL [25]	FFTformer [21]	CU-mamba [7]	EVSSM (ours)
PSNR (dB)	33.08	33.57	33.71	33.93	34.21	33.53	34.50
SSIM	0.9624	0.9656	0.9668	0.9680	0.9692	0.9650	0.9712

Figure 75: Quantitative results on the GoPro dataset.

Qualitative results

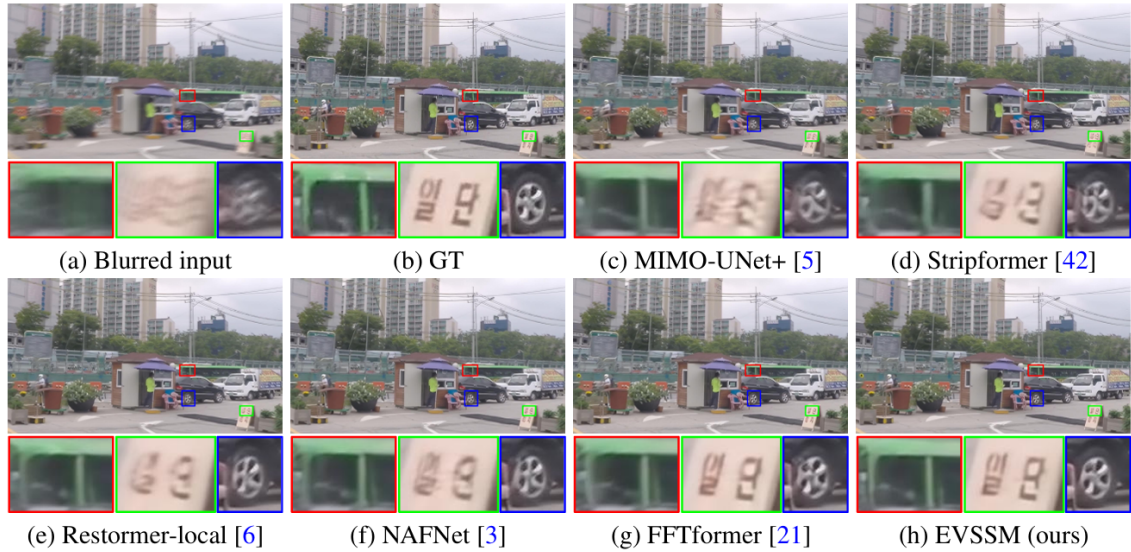


Figure 76: Qualitative results on the GoPro dataset.

4 UNet-based with Degradation representation learning

In contrast to pure end-to-end UNet-based architectures discussed in the previous section, this category of methods incorporates degradation representation learning to guide the deblurring process. While conventional end-to-end approaches may perform well on seen domains, they often suffer from overfitting and poor generalization to unseen degradation types or image distributions.

To address this limitation, these methods integrate prior knowledge of the degradation process into the deblurring model. Specifically, a UNet-style architecture is used as the primary backbone for image restoration, while a parallel branch or module is introduced to learn and encode degradation representations.

This section presents representative approaches that leverage different paradigms to model degradation: (1) CNN-based representation learning, (2) GAN-based adversarial degradation modeling, (3) Flow-based invertible degradation modeling, and (4) Diffusion-based generative degradation priors. By explicitly modeling the degradation, these hybrid frameworks aim to improve robustness and interpretability in real-world image deblurring scenarios.

4.1 CNN-based

4.1.1 Frequency-domain Learning with Kernel Prior for Blind Image Deblurring [14]

This paper proposes a UNet-based deblurring framework that addresses a key limitation in many existing CNN- and Transformer-based methods—over-reliance on training datasets for direct regression-based mapping, which can lead to severe performance degradation on out-of-domain inputs. To mitigate this, the authors introduce a Kernel Estimation Module (KE) that learns blur kernel priors from the input blurry image. This kernel prior serves as a degradation representation, enhancing the network’s generalization and reducing its dependence on training distribution. In addition, while traditional and deep learning-based approaches have largely underutilized the estimated blur kernels in the deblurring process, this work further proposes a Frequency Integration Module (FIM) to fuse the kernel prior with deep features in the frequency domain.

Specifically, there are three important concepts in this paper: (1) The Kernel Estimation Module estimates a spatially-varying blur kernel at each pixel, which is downsampled to match the resolution of the multi-scale features within UNet. (2) The FIM then uses cross-attention in the frequency domain to integrate the kernel prior and feature representations. This allows for both global information flow and kernel-guided feature enhancement. (3) A Frequency Transformer Module (FTM) is employed as the feed-forward layer, leveraging a variant of FFTformer’s Discriminative Frequency-domain Feed-forward Network (DFFN) to enhance frequency-specific feature modulation.

This architecture enables the model to dynamically adapt to varying blur patterns, improving performance on both synthetic and real-world blur scenarios that is shown on (fig.77).

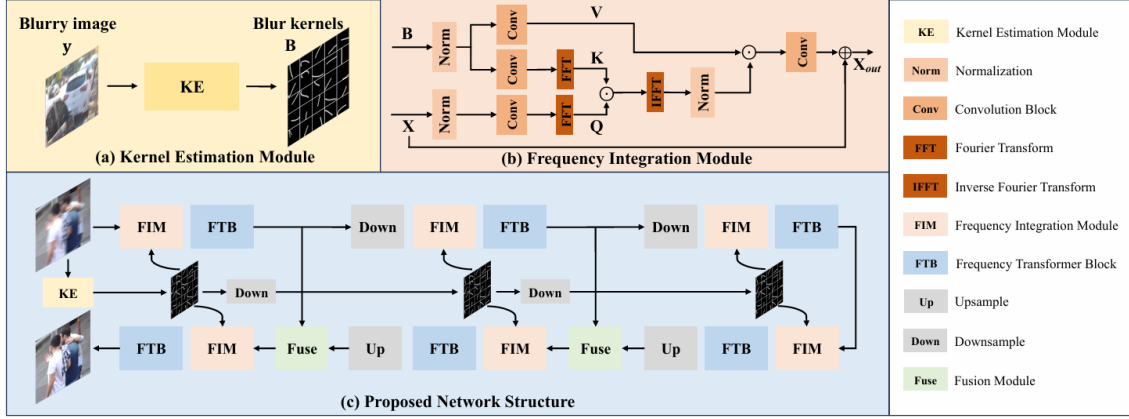


Figure 77: Overview of the network architecture of our proposed method. (a) The structure of the Kernel Estimation Module. (b) The structure of our proposed Frequency Integration Module (FIM). (c) The structure of the full pipeline in our proposed method.

The key contributions of this paper are as follows:

- **Frequency-domain Integration Module (FIM):** A novel module is introduced to integrate blur kernel priors and image features in the frequency domain using cross-attention, enabling more effective feature enhancement guided by degradation cues.
- **Improved Generalization via Blur Kernel Priors:** By explicitly estimating and incorporating blur kernel priors into the network, the model achieves improved generalization to out-of-domain blur patterns, addressing a common limitation in existing regression-based deblurring methods.

The Kernel Estimation Module (KE) is designed to predict spatially-varying blur kernels for each pixel in the input image. It adopts the approach introduced in KOALANet, originally proposed for super-resolution tasks. The KE module is implemented as a UNet-like architecture, denoted as $f_\theta(\cdot)$, and is trained using a self-supervised reblurring loss that encourages the estimated kernel to reconstruct the original blurry image when applied to the ground truth sharp image. The training objective is defined as (eq.60):

$$B_n = f_\theta(y_n),$$

$$L_{KE} = \frac{1}{N} \sum_{n=1}^N \|B_n \otimes x_n - y_n\|_1, \quad (60)$$

where n represents the n^{th} pair of training data, $y_n \in \mathbb{R}^{H*W*3}$ is blurry image, $x_n \in \mathbb{R}^{H*W*3}$ is ground truth, B_n is predicted kernels by $f_\theta(\cdot)$ that is the kernel estimation network with parameters θ , and $\otimes$ is reblur operation (convolution), $\|\cdot\|_1$ denotes the L_1 loss.

The Frequency Integration Module (FIM) is proposed to effectively fuse the kernel prior B and deep features X in the frequency domain using an attention-based mechanism (see fig.77b). The fusion process involves the following steps: (1) **Projection:** Both B and X are first projected to the same latent space using two consecutive spatial convolutions, including depth-wise convolution, to preserve spatial locality. (2) **Cross-Attention in Frequency Domain:** The attention computation is performed in the frequency domain using an FFT-based approach, similar to FFTformer. The attention matrix is derived from the dot product of frequency-transformed query and key features. (3) **Shortcut Connection:** A residual shortcut is added to the output of the attention mechanism to enable faster gradient propagation and better training stability.

In the training strategy, unlike previous works that train the network in an end-to-end fashion, this method employs a three-stage training pipeline:

Stage1–Train KE module:

The KE module is trained independently using the reblurring loss defined in (eq.60).

Stage2–Train UNet with fixed KE:

With the KE module frozen, the UNet is trained using both spatial and frequency-domain reconstruction losses (eq.61):

$$\begin{aligned}
L_{cont} &= \sum_{n=1}^N \frac{1}{N} \|\hat{x}_n - x_n\|_1, \\
L_{freq} &= \sum_{n=1}^N \frac{1}{N} \|\mathcal{F}(\hat{x}_n) - \mathcal{F}(x_n)\|_1, \\
L_{stage2} &= L_{cont} + \lambda L_{freq},
\end{aligned} \tag{61}$$

where $\hat{x}_n$ is predicted sharp image, x_n is ground truth, $\mathcal{F}(\cdot)$ is Fourier transform, and λ is hyperparameter.

Stage3–Fine-tune entire network:

The KE module is unfrozen, and the entire network is jointly fine-tuned with the combined objective (eq.62):

$$\begin{aligned}
B_n &= f_{\theta}(y_n), \\
L_{KE} &= \frac{1}{N} \sum_{n=1}^N \|B_n \otimes x_n - y_n\|_1, \\
L_{stage3} &= L_{cont} + \lambda_1 L_{freq} + L_{KE},
\end{aligned} \tag{62}$$

where λ_1 is hyperparameter.

Quantitative results

Method	GoPro		HIDE		RealBlur-R		RealBlur-J	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DeblurGAN-v2 [15]	29.55	0.934	26.61	0.875	35.26	0.944	28.70	0.866
SRN [31]	30.26	0.934	28.36	0.915	35.66	0.947	28.56	0.867
DMPHN [38]	31.20	0.945	29.09	0.924	35.70	0.948	28.42	0.860
SAPHN [29]	31.85	0.948	29.98	0.930	-	-	-	-
MIMO-UNet+ [7]	32.45	0.957	29.99	0.930	35.54	0.947	27.63	0.837
MPRNet [36]	32.66	0.959	30.96	0.939	35.99	0.952	28.70	0.873
DeepRFT+ [19]	33.23	0.963	31.42	0.944	35.86	0.950	28.97	0.884
Restormer [37]	32.92	0.961	31.22	0.942	36.19	0.957	28.96	0.879
HI-Net-Local [8]	33.08	0.962	30.66	0.936	-	-	-	-
Stripformer [32]	33.08	0.962	31.03	0.940	36.07	0.952	28.82	0.876
MSDI-Net [17]	33.28	0.964	31.02	0.940	35.88	0.952	28.59	0.869
MPRNet-Local [8]	33.31	0.964	31.19	0.942	-	-	-	-
HI-Diff [6]	33.33	0.964	31.46	0.945	36.28	0.958	29.15	0.890
Restormer-Local [37]	33.57	0.966	31.49	0.945	-	-	-	-
NAFNet [5]	33.71	0.967	31.32	0.943	35.50	0.953	28.32	0.857
GRL-B [18]	33.93	0.968	31.65	0.947	-	-	-	-
FFTFormer [12]	34.16	0.969	31.62	0.946	35.46	0.952	28.04	0.852
Ours	34.21	0.969	32.08	0.952	36.59	0.958	29.01	0.881

Figure 78: Quantitative results on GoPro, HIDE and RealBlur dataset.

Method	Parameters(M)	GFLOPs	Avg.runtime(s)
MIMO-UNet+ [7]	16.1	1235.3	0.02
DeepRFT+ [19]	23.0	187.0	0.09
MPRNet-local [8]	20.1	778.2	0.11
NAFNet [5]	67.8	65.0	0.04
FFTFormer [12]	16.6	131.8	0.13
Ours	24.4	250.2	0.14

Figure 79: Quantitative evaluations on computational costs. The average runtime of all methods is tested on images with the size of 256×256 pixels.

Qualitative results

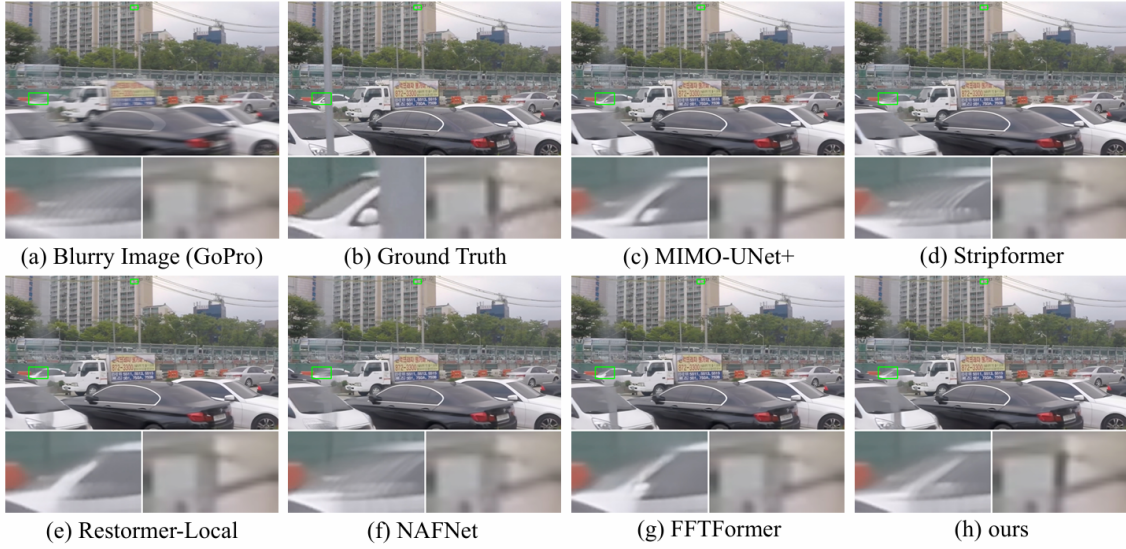


Figure 80: Qualitative results on the GoPro dataset.

4.2 GAN-based

4.2.1 Learning Degradation Representations for Image Deblurring [10]

Existing deblurring methods face fundamental limitations depending on how they model the degradation process. Traditional kernel-based approaches typically formulate the degradation as (eq.63):

$$y = F(x, k) + n, \quad (63)$$

where x is the clean image, k is the blur kernel, n is additive noise, and $F(\cdot)$ often denotes a convolution operator. These methods commonly adopt Maximum a Posteriori (MAP) estimation (eq.64):

$$(k, x) = \arg \max \mathbb{P}(y|x, k)\mathbb{P}(x)\mathbb{P}(k), \quad (64)$$

However, this formulation assumes the degradation process is uniform and convolutional, which may not hold in real-world scenarios involving complex and spatially-varying blur. As a result, the deconvolution process may produce artifacts and lead to inaccurate reconstructions. On the other hand, kernel-free approaches aim to bypass this assumption by directly learning a mapping from blurry to sharp images. However, these methods often struggle with complex and diverse blur patterns, as they lack an explicit and structured representation of degradation. Unlike noise, which can be parameterized with a scalar noise level, blur is more difficult to summarize with a simple descriptor.

Overview of the Proposed Method is to address these challenges, this work proposes a novel degradation-aware deblurring framework composed of: (1) A Degradation Encoder that extracts

a learned blur representation from the input blurry image. (2) A Reblurring Network, which regenerates the observed blur from a clean image using the learned degradation code. (3) A Deblurring UNet, which uses both the blurry image and its degradation representation to guide restoration.

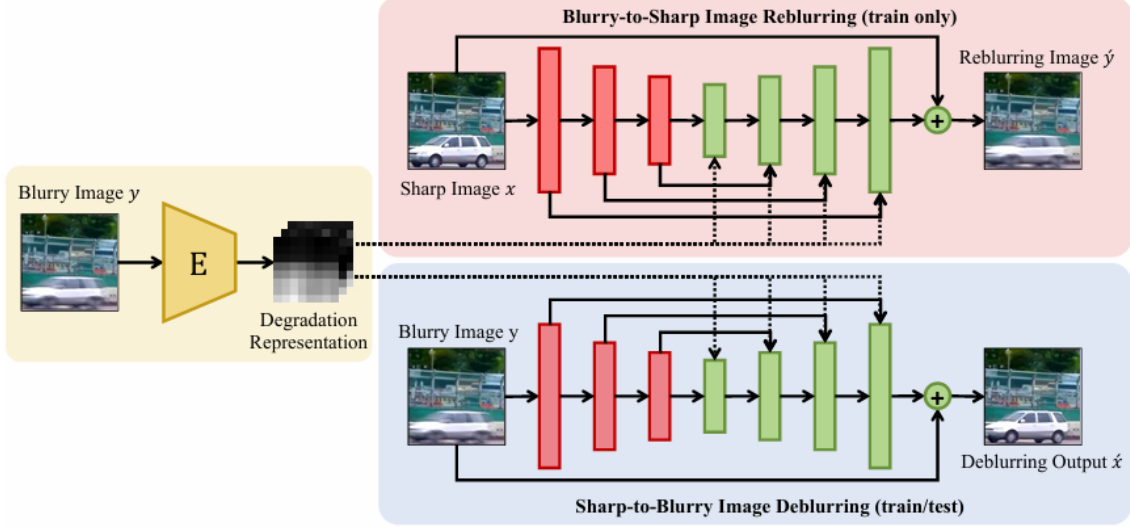


Figure 81: Learning degradation representations with reblurring and deblurring.

This design explicitly encourages the encoder to capture meaningful and discriminative degradation patterns, ensuring that the learned representation is not only compact but also functionally useful for both generation and restoration tasks. The training process jointly optimizes the encoder, reblurring, and deblurring modules in a unified framework.

This representation is then incorporated into the deblurring model as a prior. In other words, the network-based approach replaces the traditional convolution-based formulation to better handle complex real-world scenarios. The overall architecture of the deblurring model is illustrated in the following (fig.82).

There are two contributions:

- A joint framework that integrates reblurring and deblurring processes to enable the encoder to learn latent degradations within the feature space as an implicit prior, thereby enhancing the performance of the deblurring model.
- A multi-scale degradation injection network that effectively fuses the learned prior into the deblurring network to further improve restoration quality.

In this paper, there are two-Stage Training Framework. Stage 1: A joint learning framework that simultaneously trains a reblurring model and a deblurring model, enabling the encoder to learn degradation representations from blurry images. Stage 2: With the encoder parameters fixed, the deblurring model is further trained to exploit the learned degradation representations for image restoration.

Stage 1

In the first stage, a shared encoder is trained to extract degradation representations from blurry images. This encoder is jointly optimized using both a reblurring model (G_r) and a deblurring model (G_d). The goal is to ensure that the learned latent representation effectively captures the blur characteristics.

Initially, the authors experimented with an L_1 loss for training the reblurring model G_r . However, this approach yielded suboptimal results because the L_1 loss only penalizes pixel-wise differences and fails to account for perceptual similarity or structural blur patterns. To address this

limitation, the authors incorporated perceptual loss and adversarial loss as follows (eq.65):

$$\begin{aligned} L_G &= -E_{x \sim P_{data}} D(x, \hat{y}) + \lambda_1 L_{perceptual}(y, \hat{y}), \\ L_D &= -E_{(x, y) \sim p_{data}} [\min(0, -1 + D(x, y))] - E_{x \sim p_{data}} [\min(0, -1 - D(x, \hat{y}))], \end{aligned} \quad (65)$$

where $\hat{y} = G_r(E(y))$ denotes the reblurred output based on the degradation representation encoded from the input blurry image y , and $D(., .)$ denotes the discriminator used for adversarial learning.

For training the deblurring model G_d , since the degradation representation is already available from the encoder, the training focuses solely on reconstructing the sharp image. An L_1 loss is used (eq.66):

$$L(x, \hat{x}) = \lambda_2 L_1(x, \hat{x}), \quad (66)$$

Stage 2

In the second stage, the encoder is frozen, and the deblurring model is re-trained using the learned degradation representations. Following the HINet framework, the PSNR loss (eq.67) is adopted for training:

$$L(x, \hat{x}) = \text{PSNR}(x, \hat{x}) + \lambda_3 L_{blur}(x, \hat{x}) \quad (67)$$

where an additional blur-aware loss L_{blur} is introduced to emphasize blur-sensitive features. Since the encoder is specifically trained to capture blur patterns, its internal features can guide the network to generate outputs that are perceptually closer to the ground truth. The blur-aware loss is defined as:

$$L_{blur}(\hat{x}, x) = \sum_{i=1}^N \frac{1}{|E^{(i)}|} [\|E^{(i)}(\hat{x}) - E^{(i)}(x)\|_1], \quad (68)$$

where $E^{(i)}$ is i -th layer of the encoder E and $|\cdot|$ is the number of pixels in feature map.

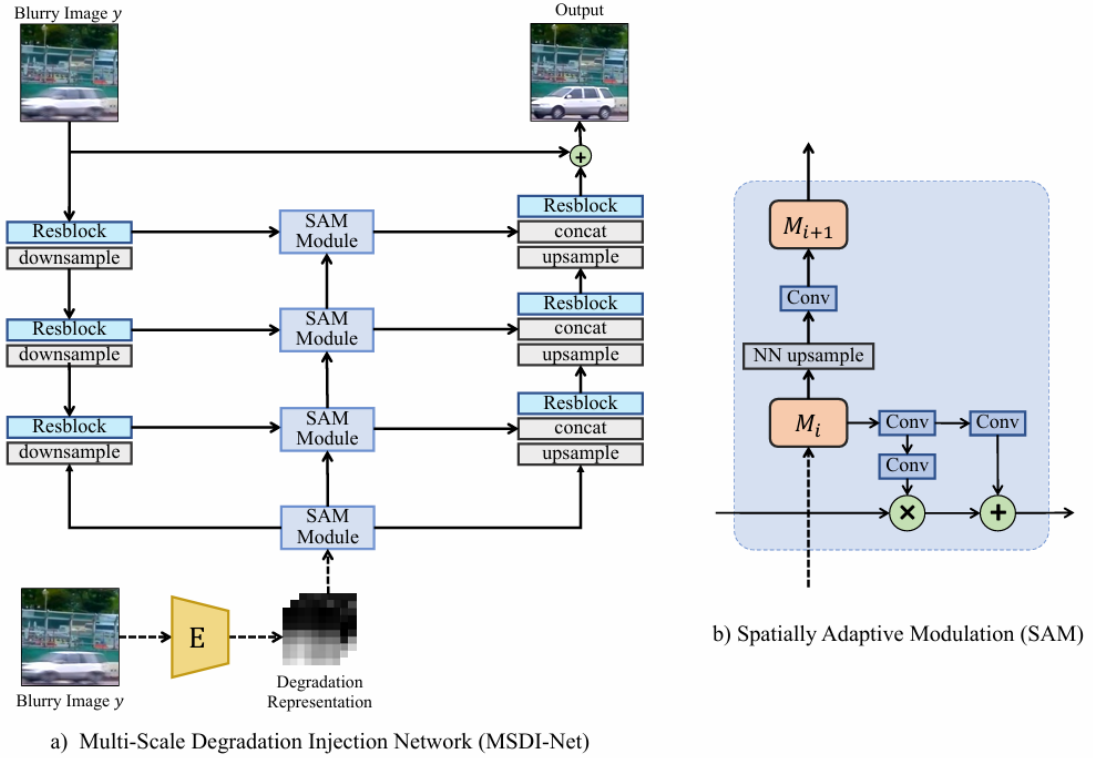


Figure 82: The network structure of multi-scale degradation injection network for image deblurring. Image reblurring shares the same structure.

The overall architecture of the proposed Multi-Scale Degradation Injection Network is illustrated in (fig.82). It consists of the following components: Encoder, Decoder Concatenation-based

Skip Connections Multi-Scale Degradation Injection Module While the encoder, decoder, and skip connections follow the typical U-Net structure, the key innovation lies in the degradation injection module. This module integrates encoder features with the learned degradation representation through Spatially Adaptive Modulation (SAM), enabling the network to adaptively incorporate degradation information at multiple scales for improved restoration performance.

Finally, the authors also investigate the properties of interpolation and decoupleness of the degradation representations.



Figure 83: Generating blurry images with linearly interpolated degradation representations. From left to right: the blurry level from sharp to blur.

Regarding interpolation, representations are extracted from both sharp and blurry images. By interpolating between these two latent representations, and subsequently applying the reblurring process on the intermediate representations, the resulting reblurred images exhibit a gradual increase in blur intensity (fig.83). This observation confirms that the learned degradation representations form a meaningful and continuous latent space that smoothly encodes varying degrees of blur.

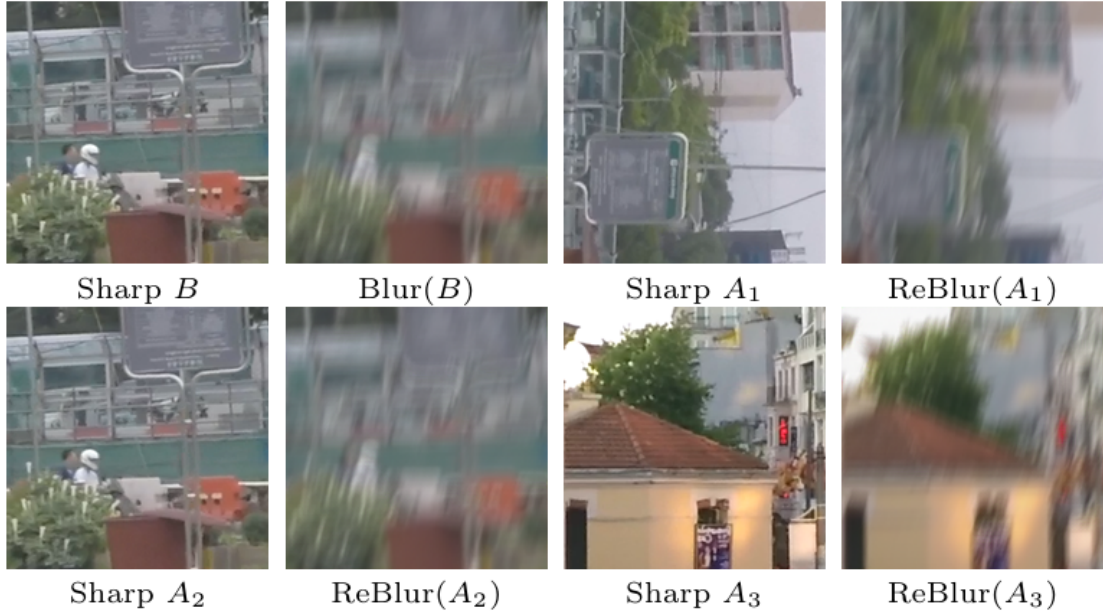


Figure 84: Reblurring A_1 , A_2 , A_3 with the degradation representation of $Blur(B)$

Regarding decoupleness, the authors evaluate whether different degradation representations alter the intrinsic contextual content of the image during the reblurring process shown in (fig.84). By applying reblurring using various degradation representations, they observe that the contextual information of the original image is largely preserved. To quantitatively assess this, the

authors employ the contextual similarity (CX) metric to measure the similarity between image pairs, with results as follows: $CX(blur(A), A) = 2.72$, $CX(ReBlur(A, deg_B), A) = 2.65$, $CX(blur(A), B) = 5.43$, and $CX(ReBlur(A, deg_B), B) = 5.39$ where A and B denote different images, deg_B is the degradation representation extracted from image B , and $ReBlur(A, deg_B)$ indicates reblurring image A using the degradation representation of B .

These results demonstrate that the degradation representations, when used with the reblurring network, can effectively replace traditional convolution-based operations without compromising the contextual integrity of the image content.

Quantitative results

Method	Reference	PSNR	SSIM
DeblurGAN [12]	CVPR'18	28.70	0.858
DeblurGAN-v2 [13]	ICCV'19	29.55	0.934
SRN [38]	CVPR'18	30.26	0.934
Gao et al. [6]	CVPR'19	30.90	0.935
DBGAN [53]	CVPR'20	31.10	0.942
MT-RNN [27]	ECCV'20	31.15	0.945
DMPHN [49]	CVPR'19	31.20	0.940
Suin et al. [36]	CVPR'20	31.85	0.948
MIMO-UNet [4]	ICCV'21	32.45	0.957
MPRNet [47]	CVPR'21	32.66	0.959
HINet [3]	CVPRW'21	32.71	0.959
MPRNet-patch256 [47]	CVPR'21	<u>32.96</u>	<u>0.961</u>
Ours	ECCV'22	33.28	0.964

Figure 85: Quantitative results on the GoPro dataset.

Method	Reference	PSNR	SSIM
DeblurGAN-v2 [13]	ICCV'19	29.69	0.870
SRN [38]	CVPR'18	31.38	0.909
MPRNet [47]	CVPR'21	31.76	<u>0.922</u>
MIMO-UNet [4]	ICCV'21	32.05	0.921
HINet [3]	CVPRW'21	<u>32.12</u>	0.921
Ours	ECCV'22	32.35	0.923

Figure 86: Quantitative results on the RealBlur dataset.

Model	Blurriest 10%	Sharpest 10%	All	MACs (G)
MPRNet-patch256 [47]	29.31	35.52	32.96	760.11
Ours	29.65	35.58	33.28	336.43

Figure 87: Detailed comparisons of MPRNet and our method on GoPro test dataset. MACs are estimated with the input size of $3 \times 256 \times 256$.

Qualitative results

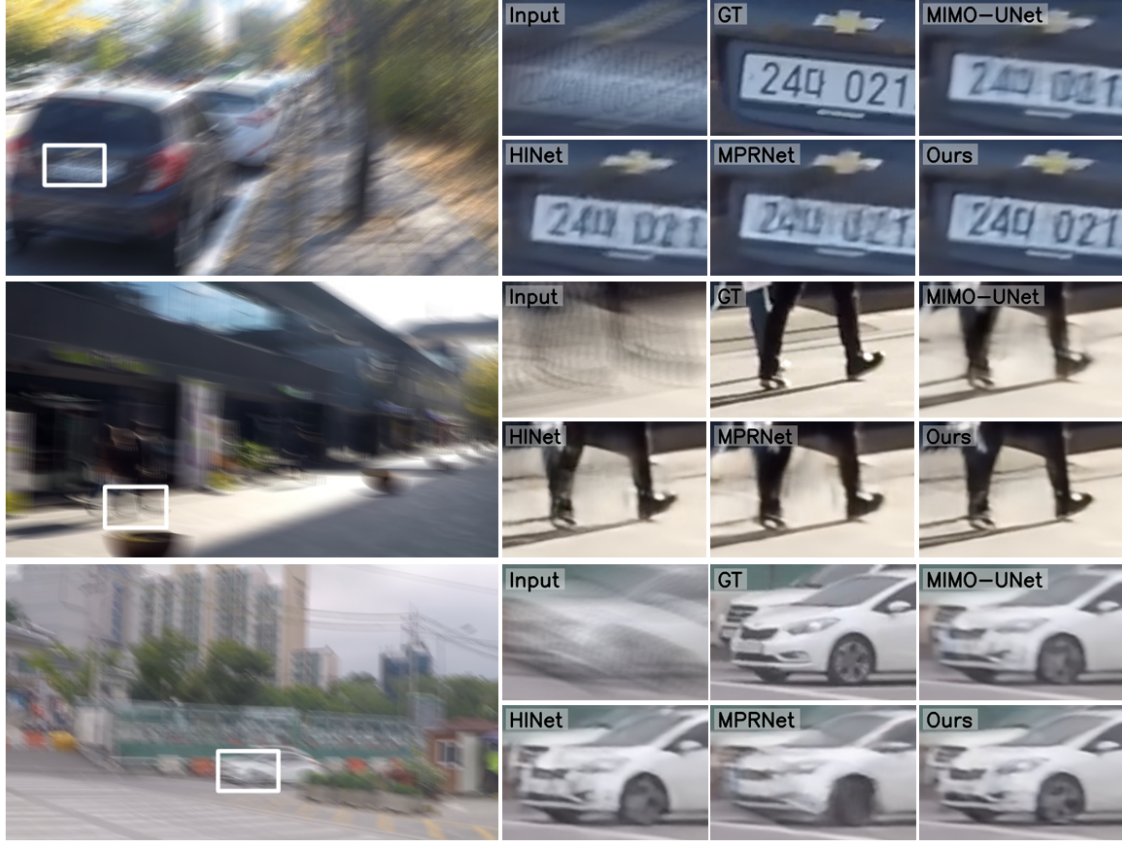


Figure 88: Qualitative results on the GoPro dataset.

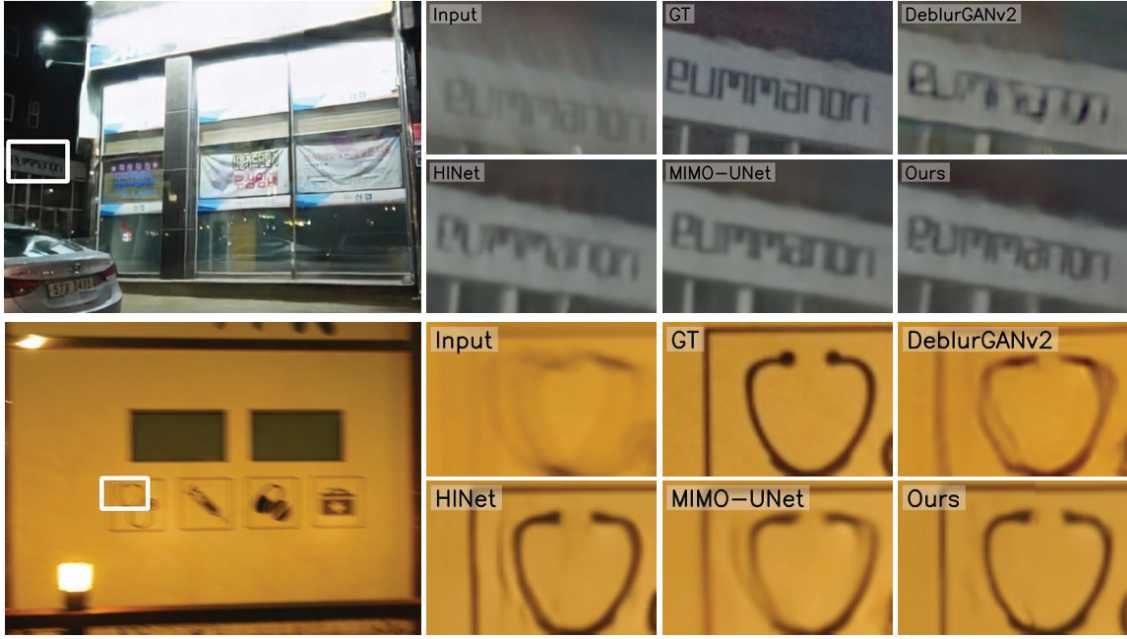


Figure 89: Qualitative results on the RealBlur dataset.

4.3 Flow-based

4.3.1 Self-supervised Non-uniform Kernel Estimation with Flow-based Motion Prior for Blind Image Deblurring [5]

Common Challenges in Image Deblurring. Image deblurring faces several fundamental challenges, including: (1) the ill-posed nature of the inverse problem, (2) non-uniform blur kernels caused by spatially varying motion, (3) the tendency of end-to-end models to overlook motion priors, and (4) the lack of real-world blurred-sharp image pairs for supervised training.

The authors address these issues as follows: To tackle the ill-posed problem, they propose an uncertainty learning framework that generates stochastic latent codes in the latent space. These latent codes are then fed into a pretrained Normalizing Flow model to estimate a motion blur kernel for each pixel, effectively solving the non-uniform blur kernel problem. Since the Normalizing Flow model is trained in a self-supervised manner, this approach also mitigates the lack of ground-truth paired datasets. Finally, the estimated motion blur kernels are integrated as prior information into existing deblurring networks (e.g., NAFNet, U-Net, CNN-based models) via a Kernel Attention Module (KAM), addressing the issue of end-to-end methods neglecting motion priors.

There are three main contributions:

- A CNN-based method to estimate spatially varying latent codes, coupled with a Normalizing Flow model to estimate pixel-wise motion blur kernels in the latent space, trained in a self-supervised fashion.
- Incorporation of uncertainty learning into the kernel estimation network to enhance both performance and robustness.
- Proposal of a fusion module to integrate the kernel estimation information into existing deblurring architectures effectively.

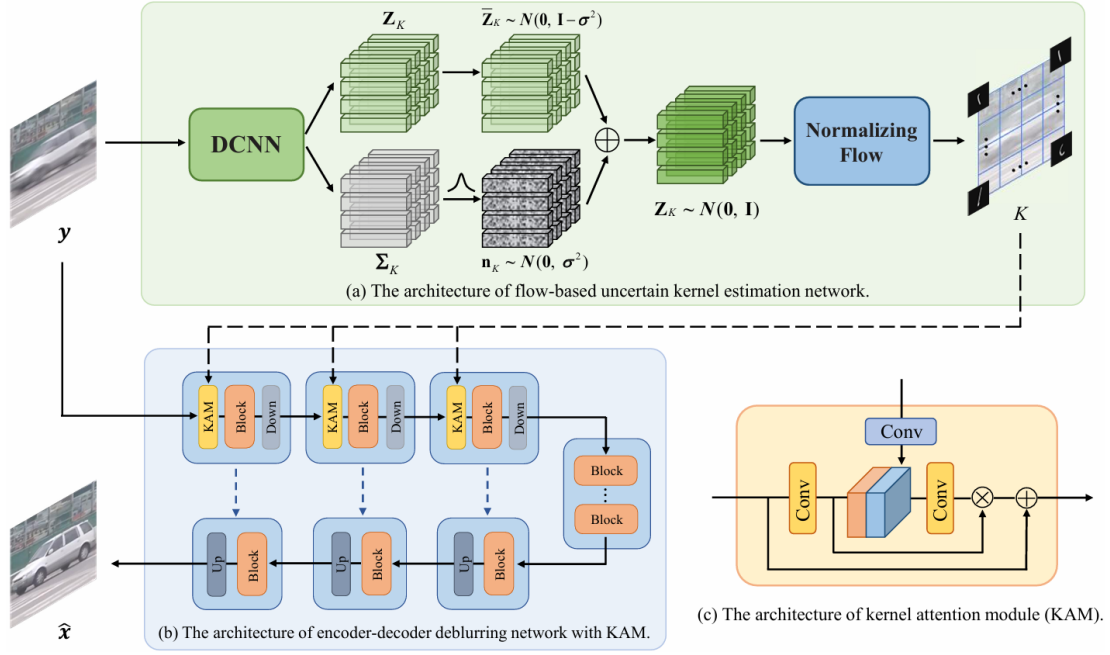


Figure 90: Overview of the proposed UFPNet for blind image deblurring. The architecture of (a) the flow-based uncertain kernel estimation network, (b) the encoder-decoder deblurring network with kernel attention module (c) the kernel attention module (KAM).

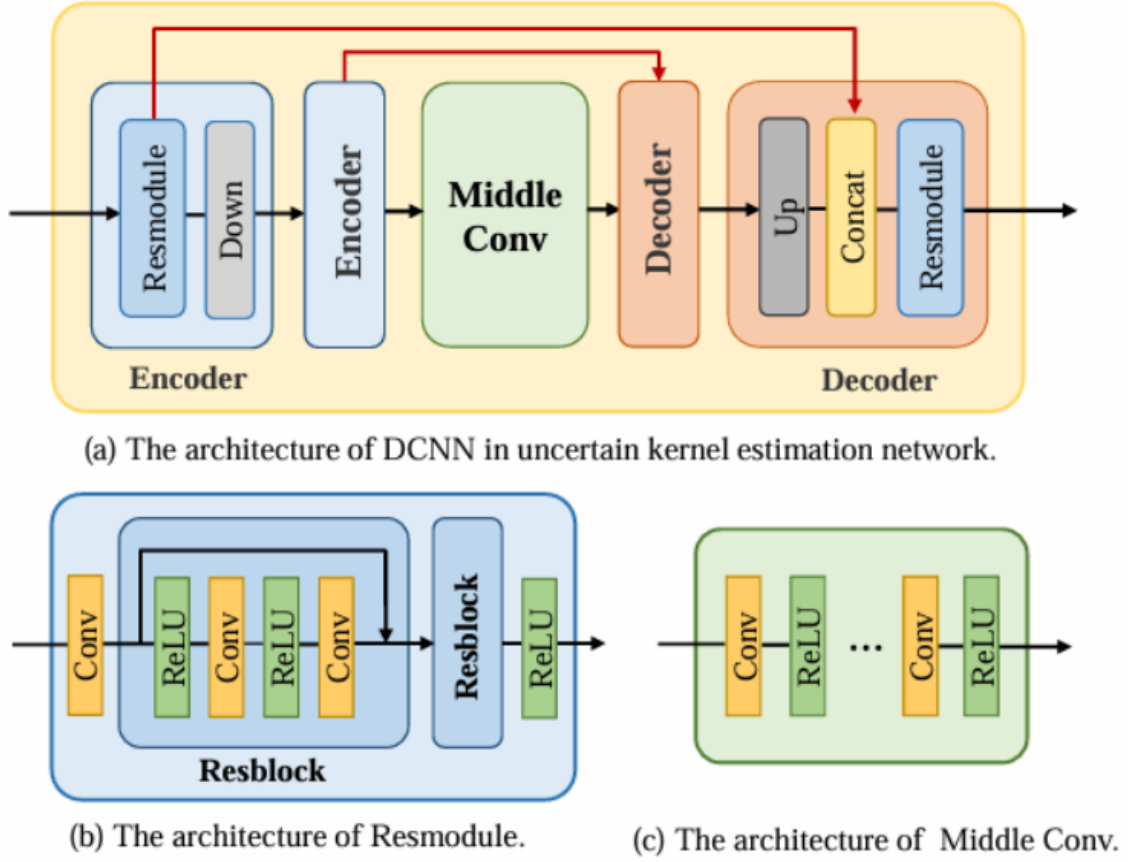


Figure 91: The architecture of (a) the DCNN in the proposed kernel estimation network. (b) the Resmodule. (c) the Middle Conv.

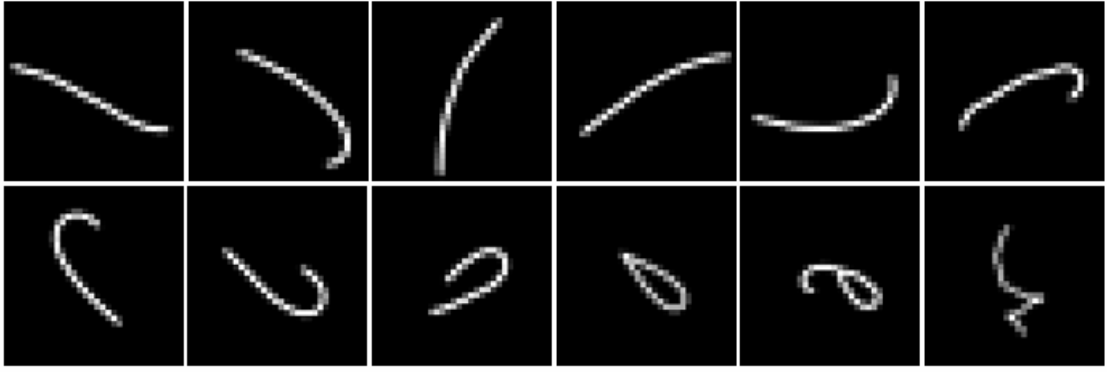
The deblurring network backbone is based on NAFNet, with the Kernel Attention Module (KAM) integrated before each layer to fuse the prior information effectively. The network architecture is shown on (fig.90).

The self-supervised kernel estimation in latent space is implemented using a Normalizing Flow model. Specifically, blur kernels for training are generated following the random trajectories method (fig.20) described in the DeblurGAN paper to construct the training dataset.

A flow model $f_\theta(\cdot)$ is then trained to learn a bijective mapping between the motion blur kernel distribution and a Gaussian distribution that is shown on (fig.92), formulated as follows (eq.69):

$$k = f_\theta(z), \quad f_\theta^{-1}(k) = z, \quad (69)$$

where $f_\theta(\cdot)$ is flow model, $\mathbf{k}$ is kernel, $\mathbf{z}$ is latent variable (Gaussian distribution)



(a) Some samples of the simulated motion blur kernels.

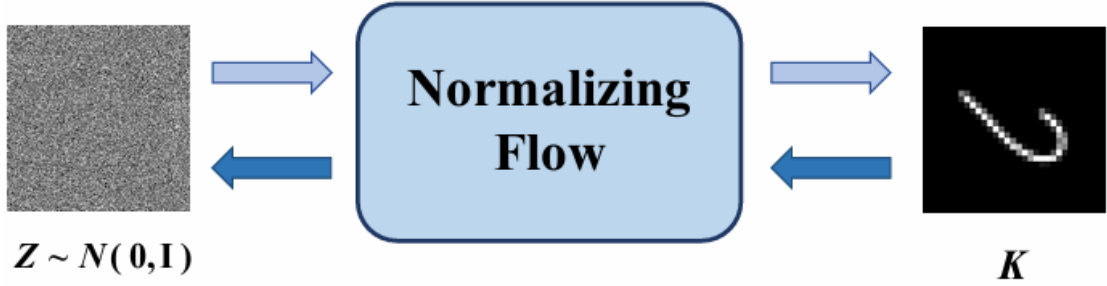


Figure 92: The illustration of the flow-based motion prior model.



Figure 93: The illustration of the normalizing flow model, each flow block which consists of a batch normalization layer, a permutation layer and a affine transformation layer.

The Uncertainty Learning as illustrated in (fig.90a), a Deep Convolutional Neural Network (DCNN) is first applied to the blurry image to predict, for each pixel i , a normalized latent code $z_i \in \mathbb{R}^{L \times 1 \times 1}$ and the corresponding standard deviation $\sigma_i \in \mathbb{R}^{L \times 1 \times 1}$, where $L = k^2$ denotes the

kernel size. For the entire image, the latent codes and standard deviations are represented as $Z_K \in \mathbb{R}^{L*H*W}$ and $\Sigma_K \in \mathbb{R}^{L*H*W}$, respectively. The core of the uncertainty modeling involves resampling Gaussian noise for each latent code to generate an uncertainty component $n_i \sim \mathcal{N}(0, \sigma_i^2)$. The latent code is then normalized as $\bar{z}_i = \sqrt{1 - \sigma_i^2} z_i \sim \mathcal{N}(0, I - \sigma_i^2)$ such that the sum of the normalized latent code $\bar{z}_i$ and the uncertainty component n_i follows a standard normal distribution. This formulation enables the subsequent use of the flow model to predict the motion blur kernel conditioned on the stochastic latent codes.

The overall design of the Kernel Attention Module is illustrated in (fig.90c). Its primary function is to fuse the estimated motion kernel information into the deblurring network, thereby enabling the model to effectively leverage the learned kernel priors during image restoration.

The training process of normalizing flow model using normalizing flow loss (eq.70) to map the motion blur kernels to a Gaussian distribution.

$$\mathcal{L}(k; \theta) = -\log p_Z(f_\theta(k)) - \log |det(\frac{\partial f_\theta(k)}{\partial k})|, \quad (70)$$

where $\frac{\partial f_\theta(k)}{\partial k}$ is Jacobian matrix of flow model, and p_Z is simple distribution(Gaussian distribution).

Training the kernel estimation network with kernel estimation network loss (eq.71) to estimate pixel-wise motion blur kernels.

$$\mathcal{L}_{KE} = \frac{1}{N} \sum_{n=1}^N \|x_n \otimes f_\theta[G(y_n)] - y_n\|_1, \quad (71)$$

where x_n is sharp images, y_n is blurry images, f_θ is flow model, and $G(y_n)$ is latent code.

Training the entire deblurring model with overall loss (eq.72).

$$\mathcal{L}_{recon} = \mathcal{L}_{PSNR} + \lambda \mathcal{L}_{reblur}, \quad (72)$$

for Reblur loss:

$$\mathcal{L}_{reblur} = \frac{1}{N} \sum_{n=1}^N \|\mathcal{F}(y_n) \otimes \mathcal{K}(y_n) - y_n\|_1, \quad (73)$$

where $\mathcal{F}(y_n)$ is reconstructed image, $\mathcal{K}(y_n)$ is blur kernels by pre-trained kernel estimation network, and y_n is blurry image.

Quantitative results

Method	GoPro		HIDE		RealBlur-R		RealBlur-J		Params (M)
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	
DeepDeblur [27]	29.23	0.916	N/A	N/A	32.51	0.841	27.87	0.827	11.7
SRN [38]	30.26	0.934	28.36	0.915	35.66	0.947	28.56	0.867	6.8
DeblurGAN [19]	28.70	0.858	24.51	0.871	33.79	0.903	27.97	0.834	N/A
DeblurGAN-v2 [20]	29.55	0.934	26.61	0.875	35.26	0.944	28.70	0.866	60.9
DBGAN [49]	31.10	0.942	28.94	0.915	N/A	N/A	N/A	N/A	11.6
DMPHN [48]	31.20	0.945	29.09	0.924	35.70	0.948	28.42	0.860	21.7
MT-RNN [31]	31.15	0.945	29.15	0.918	N/A	N/A	N/A	N/A	2.6
SAPHN [36]	31.85	0.948	29.98	0.930	N/A	N/A	N/A	N/A	23.0
MIMO-UNet [7]	32.45	0.957	29.99	0.930	35.54	0.947	27.63	0.837	16.1
MPRNet [47]	32.66	0.959	30.96	0.939	35.99	0.952	28.70	0.873	20.1
HINet [5]	32.71	0.959	30.32	0.932	35.75	0.949	28.17	0.849	88.7
DeepRFT [26]	33.23	0.963	31.42	0.944	35.86	0.950	28.97	0.884	23.0
Stripformer [39]	33.08	0.962	31.03	0.940	36.07	0.952	28.82	0.876	20.0
MSDI-Net [22]	33.28	0.964	31.02	0.940	35.88	0.952	28.59	0.869	135.4
NAFNet [4]	33.69	0.967	31.32	0.943	35.50	0.953	28.32	0.857	67.8
UFPNet (ours)	34.06	0.968	31.74	0.947	36.25	0.953	29.87	0.884	80.3

Figure 94: Quantitative results on GoPro, HIDE, and RealBlur datasets.

Qualitative results



Figure 95: Qualitative results on the HIDE dataset.

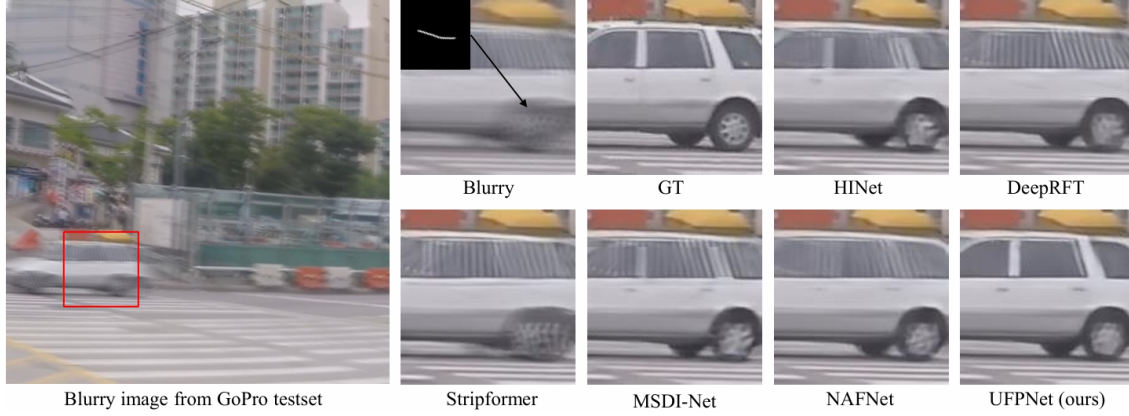


Figure 96: Qualitative results on the GoPro dataset.

4.4 Diffusion-based

4.4.1 Hierarchical Integration Diffusion Model for Realistic Image Deblurring [3]

Because most existing works rely on CNN-based or Transformer-based regression methods, which tend to recover fewer fine details due to the conservative nature of regression losses on high-frequency components. This paper’s main model is a Transformer-based U-Net architecture (Transformer) complemented by a Diffusion Model (DM) used for prior learning, it is shown on (fig.97a) and (fig.97b).

The choice of diffusion models over GANs is motivated by the fact that, unlike GANs, DMs avoid issues such as training instability and mode collapse while still generating high-fidelity images. Although diffusion models (DMs) can generate higher-fidelity details, their iterative inference process incurs a high computational cost. Moreover, DMs sometimes produce undesired artifacts that do not exist in the original clean images.

To optimize the DM-based approach, the authors employ several strategies: (1) Leveraging the latent space representation to reduce the high computational burden. (2) Integrating DM with the Transformer-based regression model to maintain distortion accuracy. (3) Adopting a hierarchical approach to effectively handle spatially varying (non-uniform) blur. A module named HIM is proposed to fuse the learned prior with the intermediate features inside the Transformer network.

There are three contributions in this paper:

- Proposed a Hierarchical Integration Diffusion Model for image deblurring.
- Applied the diffusion model in the latent space to effectively generate informative priors.
- Designed a Hierarchical Integration Module (HIM) to fuse the learned prior into the Transformer network.

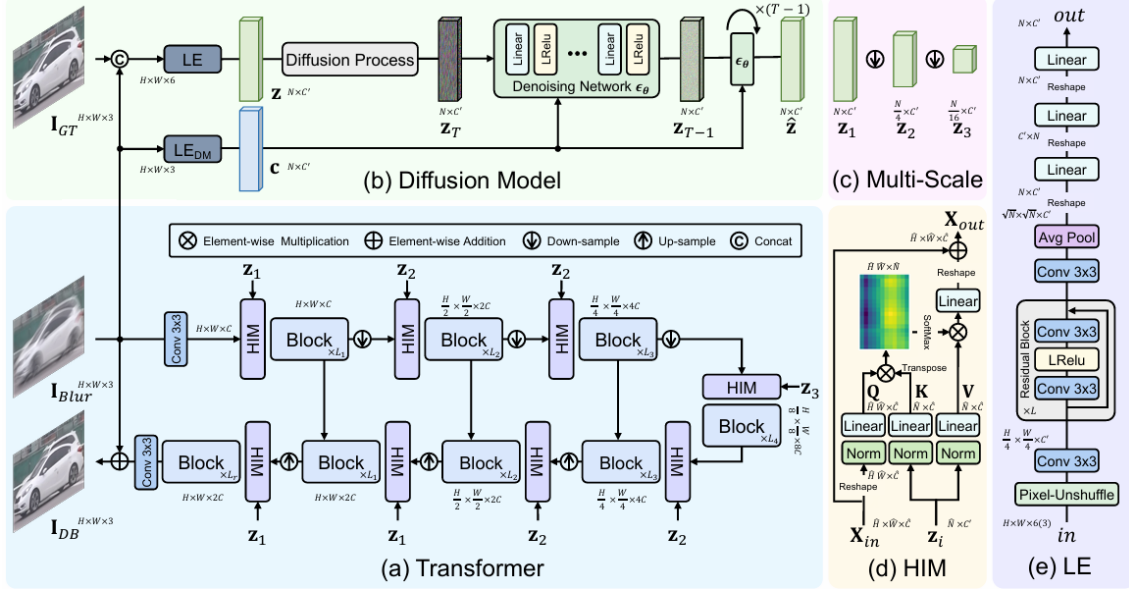


Figure 97: Proposed model

The Latent Encoder (fig.97e) is designed to compress the input image into a compact latent space representation, thereby significantly reducing the computational burden of the diffusion model.

The Hierarchical Integration Module (fig.97d) employs cross-attention mechanisms to integrate the prior features into the Transformer network effectively.

The training process is divided into two stages, as illustrated in the (fig.98): In stage one, train the Transformer and the latent encoder (LE) jointly. In stage two, freeze the latent encoder, then train the diffusion model together with the Transformer to further enhance performance.

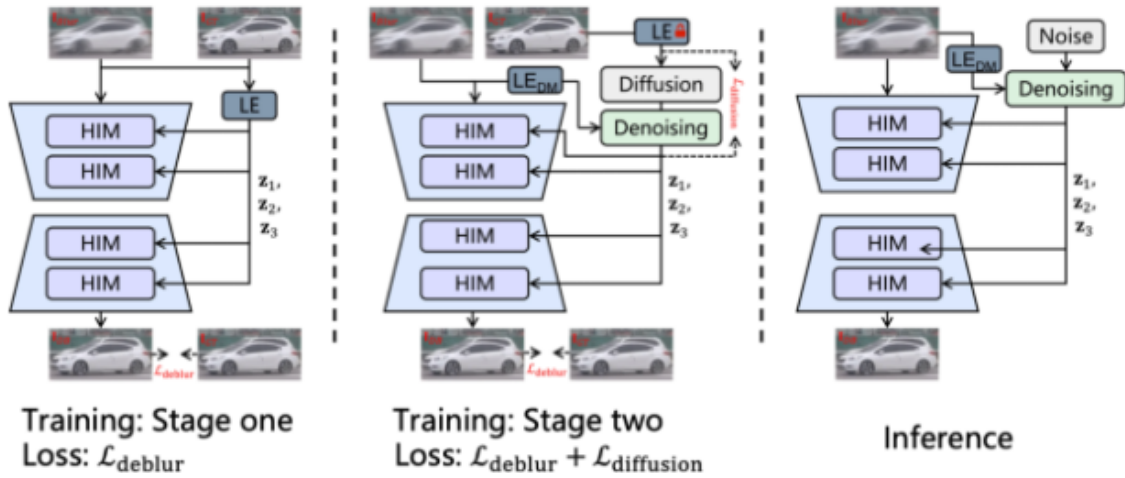


Figure 98: Training stages.

Stage One: Latent Compression

In the first stage, the Transformer and the Latent Encoder (LE) are trained jointly using the

deblurring loss L_{deblur} that is defined as (74):

$$L_{deblur} = \|I_{DB} - I_{GT}\|_1, \quad (74)$$

where I_{DB} is deblurred image, I_{GT} is ground truth.

The objective is to ensure that the LE can effectively compress the ground truth image into a meaningful prior that benefits the Transformer. The latent features produced by the LE are then downsampled and integrated into the Transformer at multiple scales, as illustrated in the (fig.97c).

Stage Two: Latent Diffusion Model

In the second stage, the parameters of the Latent Encoder (LE), trained in Stage One, are frozen. A separate latent encoder for the diffusion model, denoted as LE_{DM} , is introduced to train the diffusion model and obtain a denoising network. During this stage, the Transformer is trained jointly with the diffusion model. The training objective now includes both the deblurring loss L_{deblur} and the diffusion loss $L_{diffusion}$ that is defined as (75):

$$L = L_{deblur} + L_{diffusion}, \quad (75)$$

where $L_{diffusion} = \|\hat{z} - z\|_1$, $\hat{z}$ is predicted prior feature, z is prior feature, which helps mitigate potential mismatches between the Transformer and the diffusion model.

The key idea behind introducing the diffusion model at this stage is to replace the ground-truth latent codes used in Stage One with noise-driven samples, enabling training without relying on ground-truth latent inputs.

Quantitative results

Method	GoPro [28]		HIDE [39]		RealBlur-R [34]		RealBlur-J [34]	
	PSNR $\uparrow$	SSIM $\uparrow$	PSNR $\uparrow$	SSIM $\uparrow$	PSNR $\uparrow$	SSIM $\uparrow$	PSNR $\uparrow$	SSIM $\uparrow$
DeblurGAN [22]	28.70	0.858	24.51	0.871	33.79	0.903	27.97	0.834
DeepDeblur [28]	29.08	0.914	25.73	0.874	32.51	0.841	27.87	0.827
DeblurGAN-v2 [23]	29.55	0.934	26.61	0.875	35.26	0.944	28.70	0.866
SRN [43]	30.26	0.934	28.36	0.915	35.66	0.947	28.56	0.867
DBGAN [56]	31.10	0.942	28.94	0.915	33.78	0.909	24.93	0.745
MT-RNN [30]	31.15	0.945	29.15	0.918	35.79	0.951	28.44	0.862
DMPHN [55]	31.20	0.940	29.09	0.924	35.70	0.948	28.42	0.860
SAPHN [42]	31.85	0.948	29.98	0.930	N/A	N/A	N/A	N/A
SPAIR [32]	32.06	0.953	30.29	0.931	N/A	N/A	28.81	0.875
MIMO-UNet+ [5]	32.45	0.957	29.99	0.930	35.54	0.947	27.63	0.837
TTFA [4]	32.50	0.958	30.55	0.935	N/A	N/A	N/A	N/A
MPRNet [54]	32.66	0.959	30.96	0.939	35.99	0.952	28.70	0.873
HiNet [2]	32.71	0.959	30.32	0.932	35.75	0.949	28.17	0.849
Restormer [53]	32.92	0.961	31.22	0.942	36.19	0.957	28.96	0.879
Stripformer [44]	33.08	0.962	31.03	0.940	36.08	0.954	28.82	0.876
HI-Diff (ours)	33.33	0.964	31.46	0.945	36.28	0.958	29.15	0.890

Figure 99: Quantitative results on the GoPro, HIDE and RealBlur datasets. All models are trained only on GoPro dataset.

Dataset	Method	DeblurGAN-v2 [23]	SRN [43]	MIMO-UNet+ [5]	MPRNet [54]	BANet [45]	Stripformer [44]	HI-Diff (ours)
RealBlur-R [34]	PSNR $\uparrow$	36.44	38.65	N/A	39.31	39.55	39.84	41.01
	SSIM $\uparrow$	0.935	0.965	N/A	0.972	0.971	0.974	0.978
RealBlur-J [34]	PSNR $\uparrow$	29.69	31.38	31.92	31.76	32.00	32.48	33.70
	SSIM $\uparrow$	0.870	0.909	0.919	0.922	0.9230	0.929	0.941

Figure 100: Quantitative results on the RealBlur dataset. All models are trained and tested on the corresponding datasets.

Method	DMPHN [55]	MIMO-UNet+ [5]	MPRNet [54]	HINet [2]	Restormer [53]	Stripformer [44]	HI-Diff (ours)	HI-Diff-2 (ours)
Params (M)	21.70	16.11	20.13	88.67	26.13	19.71	28.49	23.99
FLOPs (G)	195.44	150.68	760.02	67.51	154.88	155.03	142.62	125.47
PSNR (dB)	31.20	32.45	32.66	32.71	32.92	33.08	33.33	33.28

Figure 101: Model complexity comparisons. Params, FLOPs, and PSNR on GoPro are reported. When calculating the FLOPs, the image size is set as $3 \times 256 \times 256$.

Qualitative results

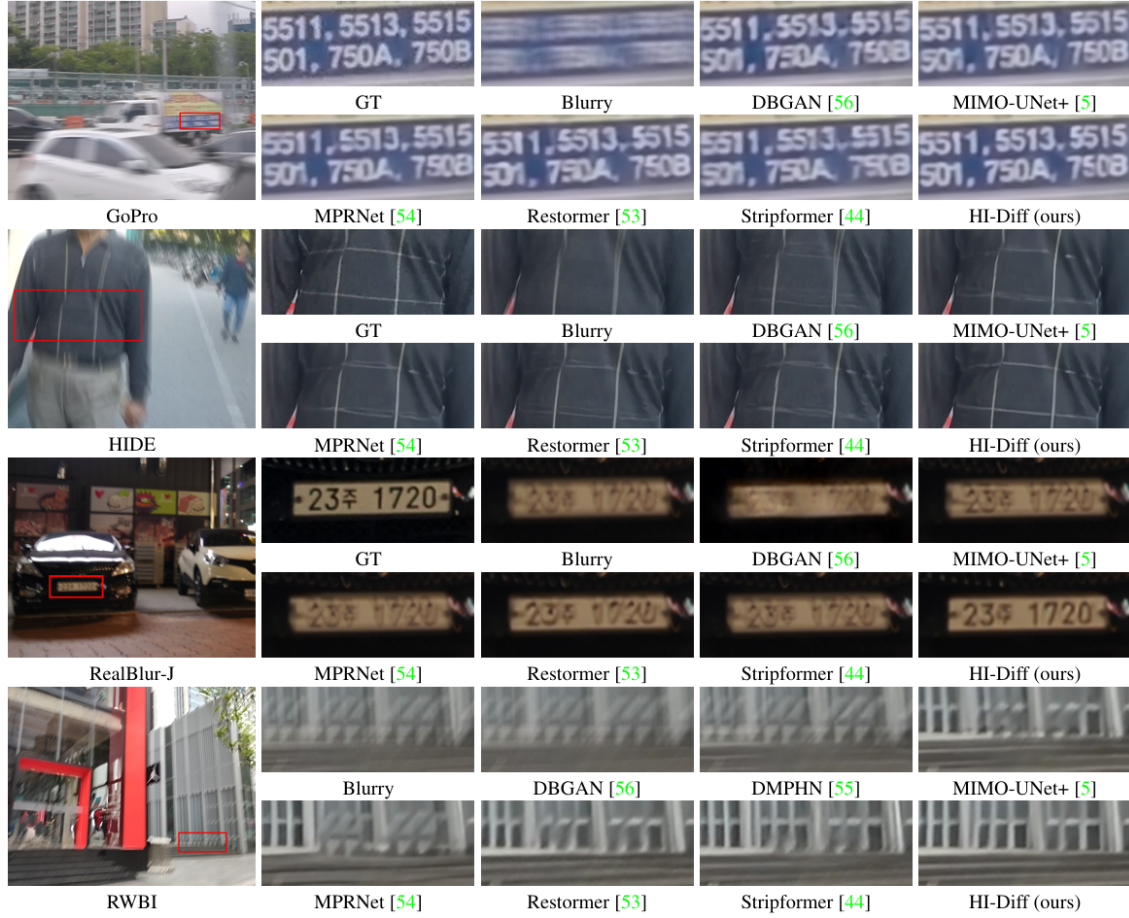


Figure 102: Qualitative results on the GoPro, HIDE, RealBlur, and RWBI datasets. Models are trained only on the GoPro dataset.

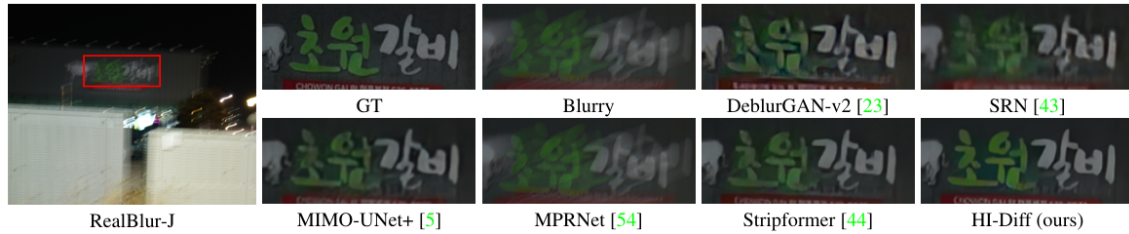


Figure 103: Qualitative results on the RealBlur dataset. Models are trained on the RealBlur dataset.

5 Dataset [17]

The following (fig.104) summarizes several commonly used datasets for image deblurring, which can be broadly categorized into synthetic and real-world datasets.

Real/Syn.	Dataset	Generative approach	Sizes	Train/valid/test
Synthetic	Köhler at al. [148]	motion trajectory captured by 6D camera	4 sharp+48 blur	Not divided
	GoPro [79]	averaging over consecutive clear frames	3,214 pairs	2,103/0/1,111
	HIDE [149]	averaging over consecutive clear frames	8,422 pairs	6,397/0/2,025
	Blur-DVS [150]	averaging over consecutive clear frames	2178 pairs	1782/0/396
Real	RealBlur [151]	blurred images: low shutter speed cameras clear images: high shutter speed cameras	4,738 pairs	3,758/0/980
	RsBlur [152]	blurred images: long exposure camera clear images: short exposure camera	13,358 pairs	8,878/1,120/3,360
	ReLoBlur [153]	blurred images: long exposure camera clear images: short exposure time camera	2,405 pairs	2,010/0/395

Figure 104: Representative benchmark datasets for evaluating single image deblurring algorithms

6 Metrics [17]

Objective assessment methods quantitatively evaluate deblurred images by constructing mathematical models to measure restoration quality. These approaches offer advantages such as fairness, objectivity, and efficiency, as they rely on numerical computations to compare the performance of different deblurring methods. Among various evaluation metrics, the most commonly used in image restoration tasks are: (1) Mean Squared Error (MSE), (2) Peak Signal-to-Noise Ratio (PSNR), and (3) Structural Similarity Index (SSIM). These metrics provide standardized and reproducible ways to assess the fidelity of the restored images with respect to ground truth.

6.1 Mean square error (MSE)

Mean Squared Error (MSE) is a widely adopted quantitative metric that measures the average squared difference between the restored image and the corresponding ground-truth sharp image. A lower MSE value, approaching zero, indicates a smaller discrepancy between the estimated and reference images, thereby reflecting a higher fidelity in the restoration outcome. Mathematically, MSE is defined as (eq.76):

$$\text{MSE}(I, S) = \frac{1}{K} \sum_{k=0}^{K-1} (I_k - S_k)^2, \quad (76)$$

where I and S represent the reconstructed image and the real sharp image, respectively. K denotes all pixels in the image.

6.2 Peak signal noise ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) is a widely utilized metric for evaluating the performance of image deblurring algorithms. It quantifies reconstruction accuracy by computing the logarithmic ratio between the maximum possible pixel intensity and the mean squared error (MSE) between the deblurred image and the ground-truth sharp image. Formally, PSNR is defined as (eq.77):

$$\text{PSNR}(I, S) = 10 \log_{10} \left(\frac{(2^n - 1)^2}{\text{MSE}(I, S)} \right), \quad (77)$$

where n is the number of digits where the image is stored. The larger the PSNR value is, the closer the recovered image is to the original image, the better the image recovery is.

It is important to note, however, that PSNR primarily captures pixel-wise differences and may not fully reflect perceptual image quality. Therefore, it is often complemented by other evaluation metrics to provide a more comprehensive assessment of visual fidelity in image deblurring tasks.

6.3 Structural similarity index measure (SSIM)

Structural Similarity Index (SSIM) is a widely adopted metric in image quality assessment, particularly effective in tasks such as image deblurring. Unlike pixel-wise measures such as MSE or PSNR, SSIM evaluates the perceptual similarity between a restored image and its corresponding ground-truth sharp image by incorporating structural information. Specifically, SSIM assesses image similarity based on three components: luminance, contrast, and structure, thereby capturing more holistic image characteristics such as texture and structural integrity.

The SSIM index between a deblurred image I and a ground-truth image S is defined as (eq.78):

$$\text{SSIM}(I, S) = [l(I, S)]^\alpha \cdot [c(I, S)]^\beta \cdot [s(I, S)]^\gamma, \quad (78)$$

where $l(I, S)$ represents the brightness comparison of the image, $c(I, S)$ represents the contrast comparison of the image, $s(I, S)$ represents the structural comparison of the image. In engineering,

often make $\alpha = \beta = \gamma = 1$. In (78),

$$\begin{aligned} l(I, S) &= \frac{2\mu_I\mu_S + C_1}{\mu_I^2 + \mu_S^2 + C_1}, \\ c(I, S) &= \frac{2\sigma_I\sigma_S + C_2}{\sigma_I^2 + \sigma_S^2 + C_2}, \\ s(I, S) &= \frac{\sigma_{IS} + C_3}{\sigma_I\sigma_S + C_3}, \end{aligned} \tag{79}$$

where μ_I, σ_I and μ_S, σ_S represent the mean, standard deviation of the images I and S , respectively. $\sigma_{IS} = \frac{1}{K} \sum_{k=0}^{K-1} (I_k - \mu_I)(S_k - \mu_S)$ indicates the covariance of the reconstructed image I and the real sharp image S . C is a constant to avoid the denominator 0 causing system errors. Both C_1, C_2 and C_3 are constants, with values ranging from $[-1, 1]$. A higher SSIM value indicates a greater similarity between the restored image and the original clear image, signifying a better restoration outcome.

7 A Short Comment on This Tutorial

This tutorial primarily introduces several recent deep learning-based research papers on image deblurring. It covers approaches ranging from early multi-stage methods to end-to-end learning and representation learning using UNet architectures. These methods integrate a variety of techniques including CNNs, self-attention, MLPs, GANs, state space models (SSM), and diffusion models. The focus of this tutorial is to explain the core concepts behind these different approaches without delving into complex mathematical derivations. For detailed methodological explanations, readers are encouraged to consult the original papers. Finally, there may be typographical errors or mistakes in this tutorial, but the main goal is for readers to grasp the fundamental ideas. If you have any comments, suggestions, or encounter any issues, please feel free to contact me via email at r13942143@ntu.edu.tw. Thank you!

References

- [1] Liangyu Chen, Xiaojie Chu, Xiangyu Zhang, and Jian Sun. Simple baselines for image restoration. In *European conference on computer vision*, pages 17–33. Springer, 2022.
- [2] Liangyu Chen, Xin Lu, Jie Zhang, Xiaojie Chu, and Chengpeng Chen. Hinet: Half instance normalization network for image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 182–192, 2021.
- [3] Zheng Chen, Yulun Zhang, Ding Liu, Jinjin Gu, Linghe Kong, and Xin Yuan. Hierarchical integration diffusion model for realistic image deblurring. *Advances in neural information processing systems*, 36, 2024.
- [4] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, and Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4641–4650, 2021.
- [5] Zhenxuan Fang, Fangfang Wu, Weisheng Dong, Xin Li, Jinjian Wu, and Guangming Shi. Self-supervised non-uniform kernel estimation with flow-based motion prior for blind image deblurring. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18105–18114, 2023.
- [6] Lingshun Kong, Jiangxin Dong, Jianjun Ge, Mingqiang Li, and Jinshan Pan. Efficient frequency domain-based transformers for high-quality image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5886–5895, 2023.
- [7] Lingshun Kong, Jiangxin Dong, Ming-Hsuan Yang, and Jinshan Pan. Efficient visual state space model for image deblurring. *arXiv preprint arXiv:2405.14343*, 2024.
- [8] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8183–8192, 2018.
- [9] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8878–8887, 2019.
- [10] Dasong Li, Yi Zhang, Ka Chun Cheung, Xiaogang Wang, Hongwei Qin, and Hongsheng Li. Learning degradation representations for image deblurring. In *European conference on computer vision*, pages 736–753. Springer, 2022.
- [11] Xintian Mao, Yiming Liu, Fengze Liu, Qingli Li, Wei Shen, and Yan Wang. Intriguing findings of frequency selection for image deblurring. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 1905–1913, 2023.
- [12] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3883–3891, 2017.
- [13] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf. Learning to deblur. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(7):1439–1451, 2016.
- [14] Jixiang Sun, Fei Lei, Jiawei Zhang, Wenxiu Sun, and Yujiu Yang. Frequency-domain learning with kernel prior for blind image deblurring. *arXiv preprint arXiv:2504.14664*, 2025.
- [15] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8174–8182, 2018.
- [16] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxim: Multi-axis mlp for image processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5769–5780, 2022.

- [17] Yawen Xiang, Heng Zhou, Chengyang Li, Fangwei Sun, Zhongbo Li, and Yongqiang Xie. Deep learning in motion deblurring: current status, benchmarks and future prospects. *The Visual Computer*, 2024.
- [18] Li Xu, Jimmy S Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. *Advances in neural information processing systems*, 27, 2014.
- [19] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5728–5739, 2022.
- [20] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Multi-stage progressive image restoration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14821–14831, 2021.
- [21] Kaihao Zhang, Wenqi Ren, Wenhan Luo, Wei-Sheng Lai, Björn Stenger, Ming-Hsuan Yang, and Hongdong Li. Deep image deblurring: A survey. *International Journal of Computer Vision*, 130(9):2103–2130, 2022.