Double Feature Fusion network: A Progressive Learning Framework for Sharper Inpainting

Appendix:

Image restoration, including denoising, deblurring and inpainting, is one of significant tasks in computer vision. However, the existing image restoration methods only handle single type of image degradation at a time. Inspired by previous image restoration work, we propose a novel multi-stage double feature fusion network called DFFNet. Specifically, the network structure consists of two parts: image deblurring and image inpainting. In the first deblurring stage multiple features likes sharpness and edge information are extracted for further training. In the second inpainting stage, both structure reconstruction and texture synthesis methods are taken into consideration. Furthermore, multiscale feature flows in the multipath contextual attention modules benefits the filling process efficiency. In order to combine the two stages, we introduce a fusion mechanism for sharper inpainting. The synthesis loss function is introduced to avoid artifacts like color discrepancy and blurriness of existing image restoration methods remarkably. extensive experiment results on various datasets show that our method outperforms current state-of-the-art methods in generating sharper, more coherent and visually plausible inpainting results both quantitatively and qualitatively.

Deep learning methods in computer vision shows tremendous progress recently. Most of image restoration methods based on single feature extraction. However, image restoration requires multifeatured information to reconstruct the properties such as sharpness, object shapes and low-quality images are general because of the natural condition and unstabe photograghy. Thus, we propose a double feature fusion network to conquer this challenge in image inpainting which is called DFFNet. It extracts deblurring and structure features simultaneously and adopts multipath refinement framework aided by multipath contextual attention modules to restore mask region in a coarse-to-fine manner. our method restores blur and large masked images and generate sharp and complete results. which outperforms state-of-the-art combined methods for sharper and smarter inpainting. Firstly, we design a multipath refinement network which could extract multiple features for further refinement and coarse inpainting. Secondly, multi-path contextual attention modules are reconstructed for receiving edge and deblur features and finer inpainting results. Thirdly, multi-stage synthesis loss function and double feature fusion units enable the image restoration to recover more details on structure and texture. Therefore, the progressive learning framework DFFNet utilize structure and texture information and refines the background for decreasing the blurry and removing noise for sharper and smarter inpainting.

3. Model Implement

3.1 The whole framework

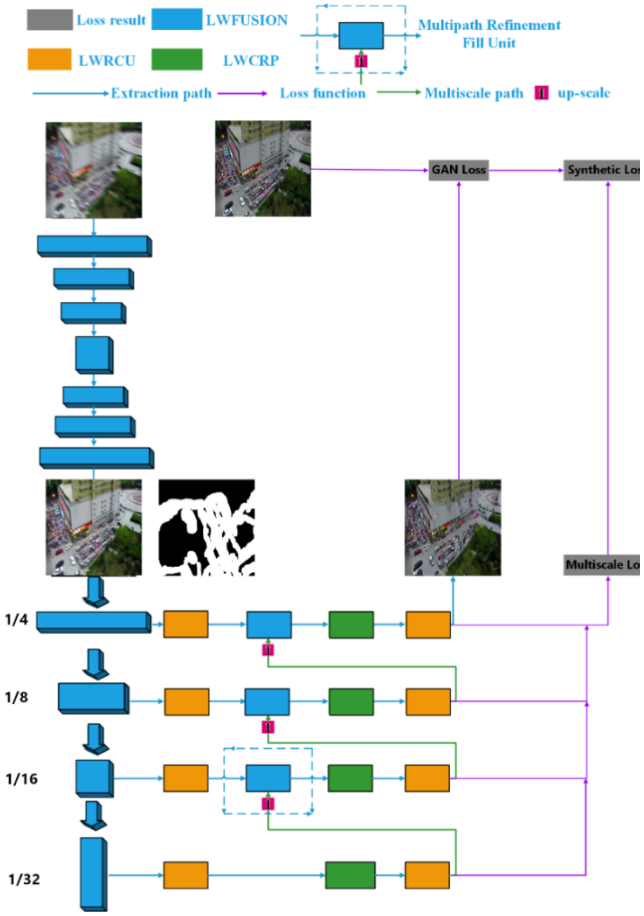


Figure 3: Whole framework of DFFNet.

In this paper, we propose a multi-stage double feature fusion network (DFFNet) to solve the problems of image deblurring and inpainting. The whole framework of DFFNet is shown in Figure 3.

To begin with, the DFFNet consists of three stages. It takes a set of blurry images, sharp images, mask images as input. In the first stage, it takes the pair of blurry and sharp images to undergo the feature extraction layers to sample the basic information of images that has been described in Section 3.2. Residual light-weight strategy and dialated convolution are used to optimal the parameters. The second stage(described in Section 3.3) refines features by extracting the feature maps of sharper and ground-truth patches. The mask images are utilized to figure out where needs to inpainting. Multipath contexual attention modules employ cohenency computation to fill in the holes. The third stage (described and illustrated in Section 3.4) fuses the sharper patches with restoration patches to generate the final prediction. The architecture and functioning of each stage of DFFNet are elaborated in the following sections.

A Network architecture

The whole framework runs in a multi-scale manner. The specific network parameters and architecture of stage one is as follows:

We first exploit the recurrent and multiscale strategies to train the network. Then, MRFNet has been extensively constructed to ensure the balance between accuracy and speed, as these aspects are currently lacking in the existing studies. Finally, a scale refinement loss function is used to train the network in a coarse-to-fine manner.

DFFNet framework for deblurring in stage one. The image is separated into different scales from top to bottom. Blue flow refers the extraction path of extracting features from scales. Then, the blocks of MRF fuse the recurrent last-round results (purple line) and the upsampling feature maps (green line) as a single refinement process. The total four refinement paths finally compute the loss in the scale refinement loss function, then the best debluring results are obtained.

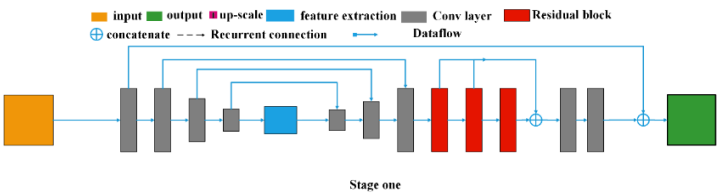
Table 6: Specific parameters of stage deblurring.

Network	Kernel	Stride	Paddin	Network	Kernel	Stride	Padding
Conv1	5×5×32	1	2	conv_r2	1×1×12	1	1
Conv2	1×1×64	1	1	conv_r2	3×3×12	1	1
Conv3	5×5×12	2	2	conv_r2	1×1×12	1	1
Conv4	1×1×12	1	1	deconv2	4×4×64	1	2
Conv5	3×3×25	1	2	conv_r3	3×3×64	1	1
Conv6	1×1×25	1	1	conv_r3	3×3×64	1	1
Conv7	3×3×25	1	2	conv_r3	1×1×64	1	1
Conv8	1×1×25	1	1	conv_r3	3×3×64	1	1
conv_r1_1	3×3×25	1	1	conv_r3	1×1×64	1	1
conv_r1_m1	3×3×25	1	1	deconv3	4×4×32	1	2
conv_r1_m2	3×1×25	1	1	conv_r4	3×3×32	1	1
conv_r1_m3	3×3×25	1	1	conv_r4	3×3×32	1	1
conv_r1_m4	3×1×25	1	1	conv_r4	3×3×32	1	1
deconv1	4×4×12	1	2	conv_r4	1×1×32	1	1
conv_r2_1	3×3×12	1	1	conv_r4	3×3×32	1	1

Table 7: Specific parameters of stage inpainting.

Network	Kernel	Stride	Padding
conv1	5×5×48	1	2
conv2	3×3×96	1	1
conv3	5×5×192	2	2
conv4	3×3×182	1	1
conv5	3×3×384	2	1
conv6	3×3×384	1	1
conv7	3×3×384	2	1
conv8	3×3×384	1	1
conv9	3×3×768	2	1
conv10	3×3×768	1	1
conv11	3×3×384	1	2
r1c1	3×3×384	1	1
r1c2	3×3×384	1	1
r1c3	3×3×384	1	1
r1c4	3×3×384	1	1
r1c5	3×3×384	1	1
r1c6	3×3×384	1	1
r1c7	4×4×192	1	2
xc1	5×5×48	1	2
xc2	3×3×48	2	1
xc3	3×3×96	1	1
xc4	3×3×96	2	1
xc5	3×3×192	1	1
xc6	3×3×192	1	1
xc7	3×3×192	1	1
xc8	3×3×192	1	1
xc9	3×3×192	1	1
xc10	3×3×192	1	1
xc11	3×3×192	1	1
xc12	3×3×192	1	1
pmc1	5×5×48	1	2
pmc2	3×3×48	2	1
pmc3	3×3×96	1	1
pmc4	3×3×192	2	1
pmc5	3×3×192	1	1
pmc6	3×3×192	1	1
pmc7	3×3×192	1	1
pmc8	3×3×192	1	1
pmc9	3×3×192	1	1
pmc10	3×3×192	1	1
pmc11	3×3×96	1	2
pmc12	3×3×96	1	1
pmc13	3×3×48	1	2
pmc14	3×3×12	1	1
pmc15	3×3×3	1	1

3.2 stage one: deblur feature extraction with residual connection



The first stage of DFFNet is designed to extract features that are required for deblurring. The input is a subset of the whole training data which only consists of blurry and ground

truth pairs. Note that (1)the overall depth of three stages are limited. (2)the subtask is deblurring and the main task is inpainting. (3)We need to fuse the refinement deblur feature for the further inpainting training.

So the first stage need to meet efficient feature extraction and a trade-off shallow depth, above all, it needs light weight and propagate features without too much pixel loss. Considering this, we designed the first stage aided only by a light weight network in thirteen layers to extract and refine patch features effectively and retain the original details remarkably. Specificlly, firstly, the images are upsampled to three scales for further refinement and fused by the U-Net, the scales are h×w, h/4×w/4×16, h/8×w/8×64, h/16×w/16×256. Secondly, after going through the UNet for encoder-and-decoder feature extraction, the patches flow into residual block to refine and retain the details of the original patches. At last, the initial extraction input and the residual refinement patches are combined into one data stream, which could be supervised by the L2 loss training loss and reconstructed into a single feature extraction model-deblurring checkpoints. Alternatively, The data stream could also be saved as the intermediate feature and ready to enhance inpainting features in stage two.

We add motion blur and Gaussian blur to simulate blur images.

We can denote a sharp image as S and a blur image as B. We find that cameras can accumulate blur artifacts by the process of exposure and capture of scenarios. This process can be described as follows:

$$B = g\left(\frac{1}{T}\int_{t=0}^{T} s(t)dt\right) \cong g\left(\frac{1}{m}\sum_{i=0}^{m-1} S[i]\right) \quad (1)$$

We explore the discrepancy of inpainting quality in various blurry images.The four-scale blur feature maps are denoted as b_k, while refinement results are denoted as l_k. First, k level of the multipath input stream concatenates same-scale feature maps b_k and upsampled feature maps l_{k+1} as middle feature maps, which is denoted as c_k.

$$c_k = b_k \oplus l_{k+1} \ (2 \le k \le 4) \quad (2)$$

Then, the fusion unit adds c_k and last-round results l_{k-1} as final results, which is denoted as l_k. This process briefly describes how refinement fusion path works. The whole process can be calculated as

$$l_k = c_k + l_{k-1} \ (2 \le k \le 4) \quad (3)$$

3.3 Stage two: inpainting feature extraction with attention module

3.3.1 Multipath feature extraction framework for inpainting

Inpainting is aided by MCA in refinement fusion paths. It can be divided into three steps. First, input features are separated into four paths by MRF, and each path has a foreground (holes prepare to fill) and a background (ready to be reshape into different scales). Each path does not need to copy several original versions. When comparing differences and computing losses, it just uses the background part as reference. Therefore, each path can extract features in different scales independently, and we can use MRF to fuse small-scale restored patches with large-scale reconstructed counterparts owing to the upsampling technique, as displayed in Figure 8.

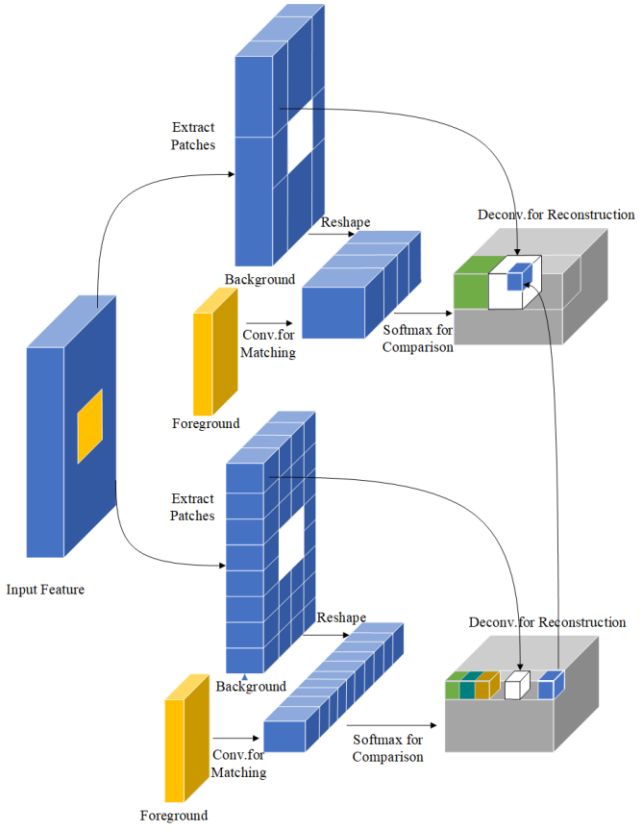


Figure 6: Graphical representation of the refinement concern layer. First, the matching value of foreground block and background block is calculated by convolution (as a convolution filter). Then, SoftMax is applied to compare and derive the attention value for each pixel. Finally, foreground patches and background patches are reconstructed by deconvolution of attention value. The context refinement layer is distinguishable and fully convoluted.

We search the input feature, extract the patches from the background, and then fold the patches into low dimensions from the background. The foreground uses coherent algorithm to compute the latent patches between assumption patches and reshape patches, and then obtain a coherent value. If the value is high, the most similar patches will be marked into an attention map with colors representing different locations.

The middle results in small scale will be reconstructed by deconvolution layer due to attention maps and SoftMax

function for comparison. Small-scale reconstructed patches upsample into the larger scale by MRF because each path obtains different scale attention and reconstructed middle results. The process can be described as follows:

$$c_i = p_i \oplus l_{i+1} \ (i = 3,4), \quad (5)$$

$$l_i = c_i + l_{i-1} \ (2 \le i \le 4), \quad (6)$$

where p_i means foreground patches, c_i represents the middle result of concatation of patch and latent refinement result, and i means the i-th path.

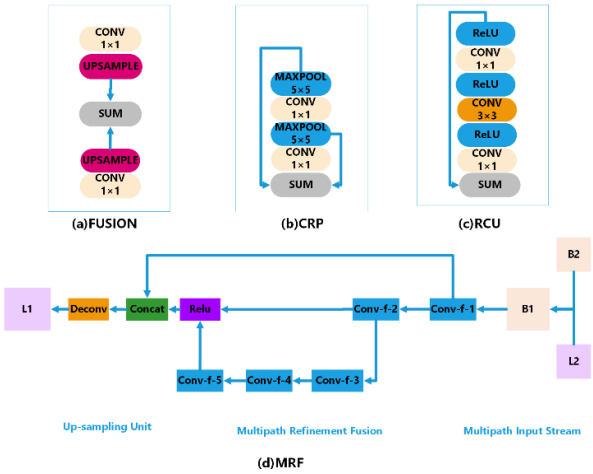


Figure. 5: Different parts of the network. (a) Fusion unit, (b) Improved CRP module, (c) Lightweight network structure of RCU, and (d) MRF unit.

Intuitively, convolution with a relatively large core size is designed to increase the size of receiving field (as well as global context coverage). The 1 × 1 convolution can only locally transform features of each pixel from one space to another. Here, we empirically prove that replacement with 1 × 1 convolution would not weaken network performance. Particularly, we replace the 3 × 3 convolution in the CRP and the fusion block with a 1 × 1 counterpart, and we modify RCU to LWRCU with a bottleneck design, as shown in Figure 7(c). Using this method, we reduce the number of parameters by more than 50% and the number of triggers by more than 75%. The convolutions are shown to reduce considerable computations without sacrificing performance.

We also enhance the MRF unit, as illustrated in Figure 7(d). Deep residual networks obtain rich feature information from multi-size inputs. Residual blocks are applied to address vanishing gradient problems. Thus, we add residual blocks to MRF.

Here, MRF is specifically designed as a combination of multiple convolution layers (conv-f-1 to conv-f-5), and each convolution layer is followed by a ReLU activation function. Conv-f-2 uses feature maps from Conv-f-1 to generate more complex feature maps. Similarly, Conv-f-4 and Conv-f-5 continue to use the feature maps generated by Conv-f-3 for further processing. Finally, the feature maps obtained from multiple paths are fused. The specific calculation expression is as follows:

$$y = f_2(f_1(x)) + f_4\left(f_3\left(f_2(f_1(x))\right)\right), \quad (4)$$

where f, x, and y represent the convolution operation, characteristic graph of input, and the characteristic graph of output, respectively.

We construct a residual connection in each path of MRF. In forward transmission, remote residual connections transmit low-level features, which are used to refine visual details of coarse high-level feature maps. The residual connections allow the gradients to propagate directly to early convolution layers, thus contributing to effective end-to-end training.

3.3.2 Attention modules and coherency algorithm

Attention map. We use specific colors to represent locations that are similar to the foreground, as shown in Figure 5. Attention maps can be best described as connections between the foreground and background.

For instance, the red area shrinks, which means finding the true area to match is difficult. Furthermore, holes would be filled with blur patches, leading to unstable image inpainting. As a result, the restored area looks blurry and twisted. Other zones encounter similar problems, and attention maps have a higher probability of recognizing wrong coherent patches. In addition, features decrease due to the blurry background, causing attention color to shrink. Thus, we need to perform a debluring process before inpainting.

Figure 4: Comparison of blur inpainting and deblur inpainting. The red attention block means the latent coherent inpainting patches are in the bottom right area, which is similar to the roads, while the pink area represents the bottom left forest. We find that the foreground of the blur image has the most possibilities of distorted shapes and wrong predicted locations.

During the training process in Figure 4, the inpainting feature seems more lighter,which means the feature has a potential tendency to aim to the no meaning areas which is represent in white. This is blurry images shows in the first row, while the second row that refines and retains the deblur patches has a vivid color which means the pixel pointing to the most similar positions with specific direction. Aided by coherency algorithm, MCA layers in stage two achieves a significant progress in matching the right structure and color for restoring and refining patches.

Coherency algorithm. The connection between p_i' and p_i can be described as follows:

$$C_{max_i} = \frac{\langle p_i, p_i' \rangle}{\|p_i\| \cdot \|p_i'\|} \quad (7)$$

As adjacent foregrounds are filled, and intuitively the most similar patch is next to element, we also need to calculate

coherent value of adjacent coherent value as follows:

$$C_{ad_i} = \frac{\langle p_i, p_{i-1} \rangle}{\|p_i\| \cdot \|p_{i-1}\|} \quad (8)$$

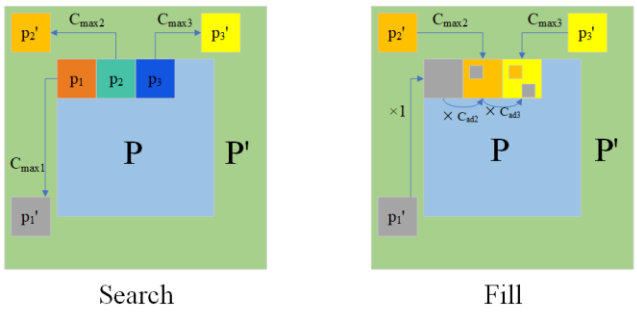


Figure 7: We define p' as background and p as foreground. First, each neural patch in the blank area P searches for the most coherent patch on the boundary p'. Then, we combine the previous generated patch and most similar contextual patch to generate the current one. This process continues for four rounds from small scale to large scale, which is called multipath contextual attention algorithm.

Figure 7 shows the patch search for the background using Eq. (7) and the search for the adjacent area using Eq. (8), and multipath makes iterative results as follows:

$$p_1 = p_1', C_{ad_i} = 0$$

$$p_i = \frac{C_{ad_i}}{C_{ad_i}+C_{max_i}} \times p_{i-1} + \frac{C_{max_i}}{C_{ad_i}+C_{max_i}} \times p_i' \; i \in (2,n) \quad (9)$$

Finally, we can conclude the process in the following algorithm:

Algorithm 1 Process of MCA layer

Input: The set of feature map for current batch F_{in}
Output: Reconstructed feature map F_{out}

Search
Reshape P' as a convolution filter and apply in P
Use Eq(1) to get the D_{maxi} and p'
Initialize p_i with m_i'
End Search
Compute
for $i \in [1,n]$ **do**
    Use Eq(8) to calculate the C_{adi}
    Use Eq(9) to get the attention map A_i for m_i
end for
Combine A_i to A_n to get a attention matrix
Reuse P' as a deconvolution to get F_{out}
End Compute
Return F_{out}
Fill
for $i \in [1,n]$ **do**
    Use the MRF to fill the hole
end for
End fill

When feature maps are filled completely (or after a specific number of refinements), feature maps have passed through MRF and MCA several times. If we directly use the last feature map to generate output, gradient vanishing could occur, and signals generated in earlier iteration are damaged. To solve this problem, we must merge intermediate feature maps. However, using convolutional operations limits the number of multipaths because the number of channels in concatenation is fixed. Directly summing all feature maps removes image details because hole regions in different feature maps are inconsistent, and prominent signals are smoothed. Thus, we use the residential remote connection to solve this problem.

Another approach is to reduce the depth of DFFNet and the network module more lightweight. On the one hand, we train the network in two parallel networks that are separate for deblurring and inpainting, and they can extract different features and run training process. On the other hand, the lightweight process also helps cut down the network size, making the connection shorter and the signals more original. MCA uses the results to compute multiscale L1 and L2 losses.

3.3.3Edge feature extraction and heatmap

The stage one of deblurring feature benefits the whole background refinement. And the refined background patches are sent into the stage two. Another feature which is ready to extracted is structure information which is vital to restore images.

The specific network structure is the same as the multipath refinement framework, the input is the ground truth images A and the masks M, the edge map B and the grayscale counterpart C. In the edge generator, we use the masked grayscale image C'=C*(1-M) as the input, so we coule get the edge map B'=B*(1-M). Then the prediction of edge map $B_{prediction}=G_e(C',B', M)$,

Then we use the ground truth B and Bprediction as the inputs of discriminator. The discriminator can compare and judge whether the edge generation is real or not. And the loss function is constrains as follows:

$$\min_{G_e} \max_{D_e} L_{G_e} = \min_{G_e} \left(a_{adv,1} \max_{D_e} (L_{adv,1}) + a_{FM} L_{AM} \right)$$

Note that the activate map loss function is the activate maps in the discriminators. It aims to force the training process to generate the patches which is similar to the original maps, it weights remains 10:1 in the real training process which is associate with adv inpainting loss function.

The edge information benefits the reconstruction of object structure. So the generative network first train the network to learn the image edge generation, and use the edge prior to predict the mask region structure. The edge feature helps out the multipath refinement fusion and multipath contextual attention alogrithms to reconstruct the images and removing occlusions simultaneously. The following is the middle feature extraction outputs for image inpainting feature extraction.

Each row consists of the deblur refined patches, edge refined feature maps and the course to fine inpainting feature results which is aided by the above two feature extraction and fusion

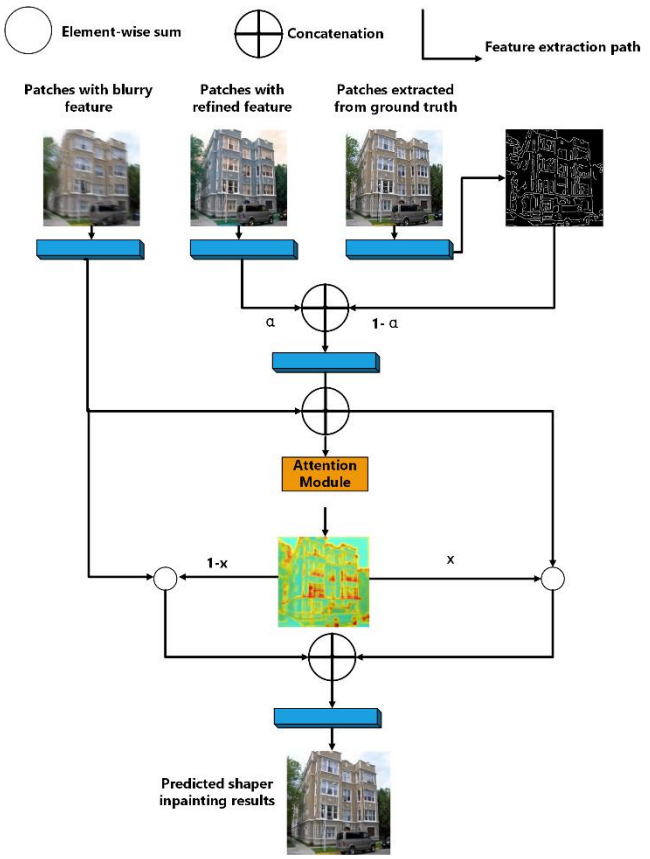
as the figure 16 illustrated in section 3.4.1.

From the effect of the still photos we remove the background occulusions and characters and restores the images in sharper inpainting.The restoration images shows vivid colors, sharp details and robust restored structures.

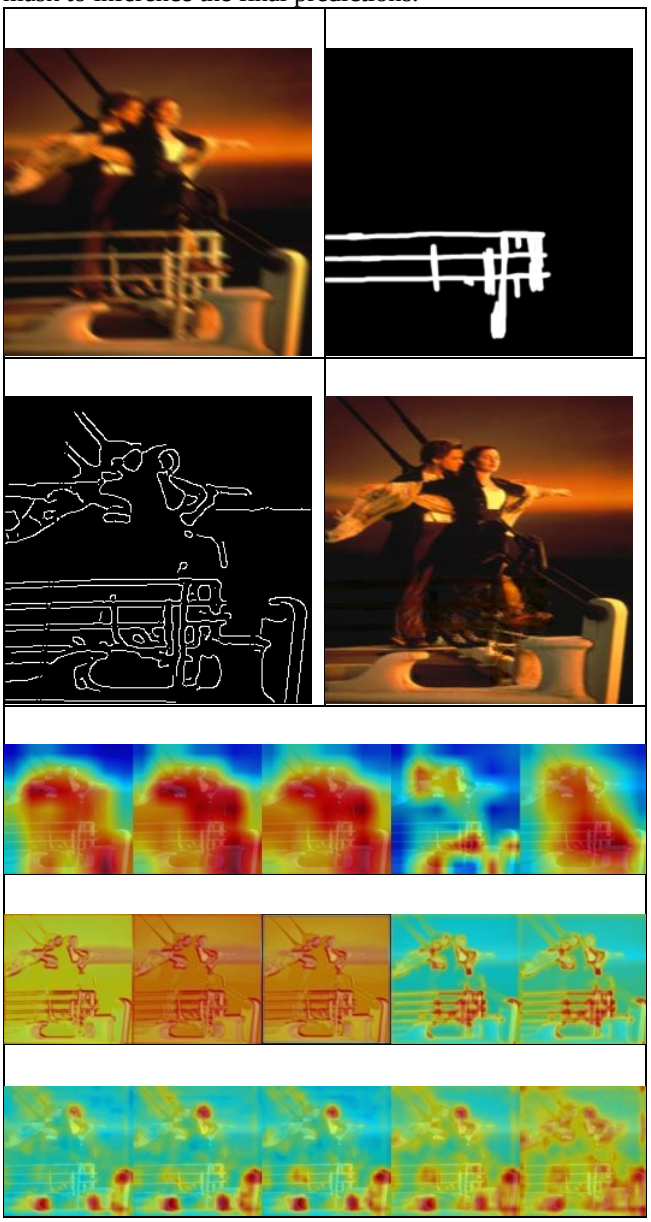
The contextual attention heatmaps which is aided by deblurring and edge reconstructed double feature fusion are generated which is shows as follows in Figure 17, and they verify the middle outputs aims to build the connection between foreground and background patches for similarity.

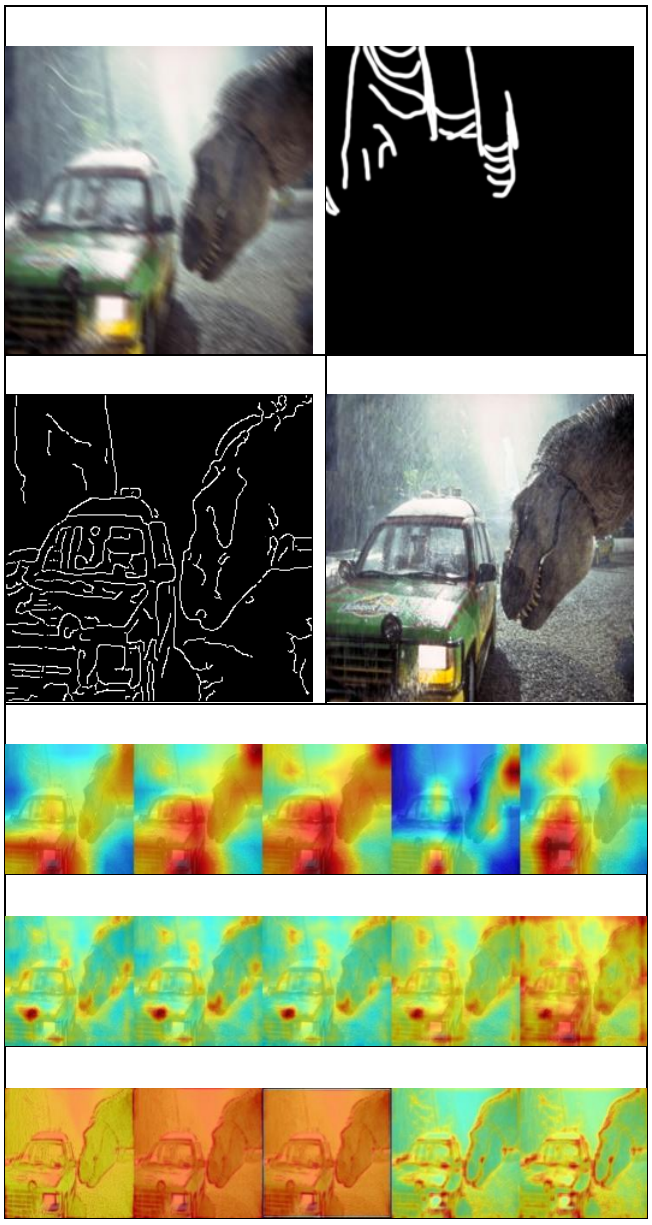
3.4 Stage three: multipath fusion with progressive weighted training

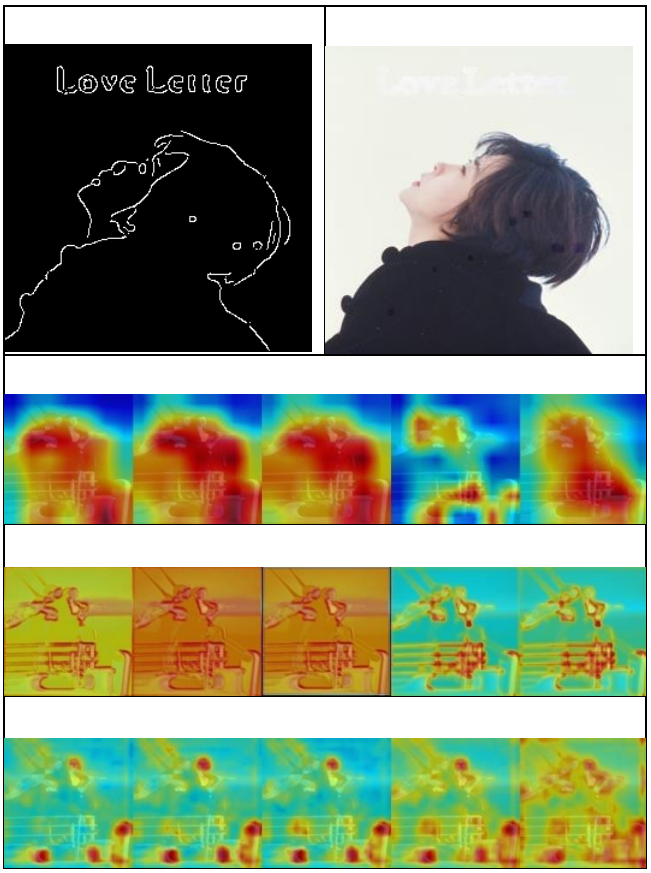
3.4.1 double feature fusion in stage three



In stage three, the task is to fuse the deblurring feature and the inpainting feature from stage one and two, to make the final restored frame. During training, the stage accepts three inputs: patches with blurry feature extracted from stage-one encoder, patches with refined feature extracted from stage-one decoder, and patches extracted from the ground truth. In the progressive weighted training process, first, the inpainting feature is extracted from the patches with refined feature and patches from ground truth, the a is a hyper parameter which is set to 0 initially to control the proportion of the refined resource. Second, the blurry patches and the mixed inpainting feature patches are combined the send to the attention module, the context attention module using the mask to predict the foreground and generate the preliminary inpainting feature. At last, the α is set to 1 and the deblur refined feature patches are sent to attention module at the middle process of training process and predicted by the attention module once again. The results are compared with the synthesis loss function between predicted sharper inpainting results and the patches with blurry feature. at the beginning of the training deblurring feature refines the input of blurry images and benefits the inpainting feature extraction, in the middle of the training process, double deblurring feature and inpainting feature make a fusion by controlling parameter a. and each path which contains different scale of double feature patches are refined and matched by multipath context attention module with the mask to inference the final predictions.







we adopt the L2 loss to restrict deblur feature by comparing the blurry patches and ground truth.

$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N} ||x_s^i - F(x_l^i)||^2, \quad (10)$$

The inpainting feature which is aided by deblur feature is trained by adversarial loss function.

$$L_{adv} = \underset{S\sim P_{sharp}}{E}\left[\log 1 - G(G(B))\right] \quad (12)$$

Due to the four paths to refine the four scale patches, each path is also restricted by the scale L2 loss function.

$$L_{scale} = \frac{1}{2K}\sum_{k=1}^{K} \frac{1}{c_k w_k h_k} ||L_k - S_k||^2, \quad (11)$$

At last, the synthesis loss function is defined as the sum of adversarial loss function and scale loss function.

$$L_{final} = L_{adv} + L_{scale} \quad (13)$$

4. Experiment

To evaluate the performance of all the strategies and architectures proposed in this work, extensive experiments have been conducted on multiple datasets. Section 4.1 and Section 4.2 shows the experiment details. Section 4.3 presents the comparative results obtained by DFFNet and other state-of-the-art techniques. Section 4.4 presents the results of the ablation studies performed over DFFNet.

4.1 training details

We implement our deblur stage by using TensorFlow. The model is trained with Adam (β1 = 0.9, β2 = 0.999). Images are randomly cropped to 256 × 256. The batch size 16 is used

for training in four NVIDIA RTX2080Ti GPUs with 11 GB. At the beginning of each epoch, the learning rate is initialized as $10-4$ and subsequently decayed by half every 10 epochs. We train 70 epochs on VisDrone and 50 epochs on Places2. The whole network DFFNet is designed for multitask of deblurring and inpainting in the meantime, and note that it could also separated into two models for plug-and-play purpose. Stage one is for deblurring models and Stage two is for inpainting models, while Stage three is for sharper inpainting in a joint training manner.

4.2. Dataset

We use two public datasets, namely, VisDrone and Places2, to train and evaluate the performance of DFFNet, and provide synthetic blur techniques on them.

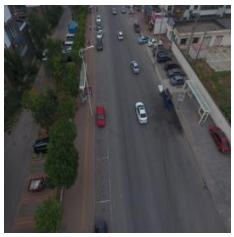
We produce VisDrone and Places2 blur dataset through different approaches. Several data augmentation techniques can be utilized to prevent our network from overfitting. In terms of geometric transformation, images are flipped horizontally, vertically, and rotated at random angles. For colors, RGB channels are replaced randomly. For image color degradation, saturation in HSV color space is multiplied by a random number in [0,5]. In addition, Gaussian blur is added to the image. Finally, standard deviation of noise is also randomly sampled from the Gaussian distribution N (0–1) to ensure that our network is robust to noise of different intensities.

4.3Compatrative experiments

4.3.1 deblur with state of the art

A. Dataset

We use two public datasets to train and evaluate the performance of DFFNet. The first dataset is VisDrone, which provides synthetic blur techniques. The second dataset is GOPRO, which is captured in real-world scenarios.



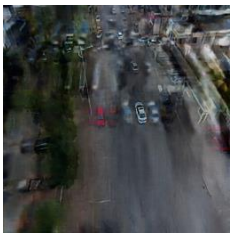
(a) Sharp image

(b)Random deviation

(c) Change blur kernel size

(d) Change shaking angle

(e) Change shaking length



(f)Motion blur

Fig. 4: Overview of dataset augmentation by changing the parameters. The first column shows a sharp image, an image blurred by changing the blur kernel size, and an image blurred by altering the blur shaking length. The second column shows the blurred image added with random standard deviation, in which the image is blurred by changing the shaking angle. The last image is generated by real motion blur.

The benchmark dataset of VisDrone consists of 288 video clips formed by 261,908 frames and 10,209 static images. The dataset, which was captured by various drone-mounted cameras, covers a wide range of elements, including location (taken from 14 different cities separated by thousands of kilometers in China), environment (urban and country), objects (pedestrian, vehicles, or bicycles.), and density (sparse and crowded scenes). The dataset was collected using various drone platforms (i.e., drones of different models) in different scenarios under various weather and lighting conditions.

We produce the VisDrone blur dataset by using different methods, including Gaussian blur. Several data enhancement techniques are utilized to prevent our network from overfitting. In terms of geometric transformation, the images are flipped horizontally, vertically, and rotated at random angles. For the colors, the RGB channels are replaced randomly. For the image color degradation, the saturation in the HSV color space is multiplied by a random number in [0,5]. In addition, Gaussian blur is added to the blurred image. Finally, to ensure that our network is robust to noise of different intensities, the standard deviation of noise is also randomly sampled from the Gaussian distribution N (0-1).

The GOPRO dataset contains images with changeable motion blurs. GOPRO cameras can record sets of sharp information that are integrated over time to generate blurred images rather than using a preset kernel to blur the generated images. When the camera sensor of GOPRO receives light during exposure, it accumulates a clear image stimulus each time, resulting in a blurred image.

B. Comparative Experiments

We conduct comparative experiments with DeblurGAN, DMPHN, and SIUN to verify the performance of our model. Our proposed DFFNet can achieve state-of-the-art performance compared with SIUN on VisDrone. The values of PSNR and SSIM are much higher than DeblurGAN and DMPHN, suggesting the advantage of our method in handling Gaussian blurs. Moreover, our method performs

better than SIUN and DMPHN and even much better than DeblurGAN in dealing with the motion blurs of GOPRO. The trends in Table II prove the superiority of the DFFNet framework based on the PSNR and SSIM values.

TABLE II. Testing results of the blurred image datasets and their PSNR and SSIM values.

Method	GOPRO		VisDrone	
	PSNR	***SSIM***	***PSNR***	***SSIM***
deblur GAN	28.22642	0.747912	28.29447	0.609642
DMPHN	34.21846	0.898285	28.54136	0.526301
SIUN	34.46135	0.900913	28.28039	0.543417
Our model	**34.63429**	**0.907881**	**29.40845**	**0.862474**

C. Ablation Experiments

The original DFF network used as the benchmark is denoted as RefineNet. From this original RefineNet, we add the lightweight process to the benchmark and denote it as LWRefineNet. Then, we add the remote residual connection to the refinement path and residual connection to fusion unit and denote it as RCRefineNet. Finally, we combine lightweight and residual strategy to the benchmark network and define it as DFFNet.

TABLE III. Memory consumption of graphics cards by different methods (Model= DFFNet).

Method	GOPRO	VisDrone
	Network(MB)+Batch(8)	***Network(MB)+Batch(16)***
deblur GAN	**4538**	6012
DMPHN	6541	7329
SIUN	8399	8561
Our model	5452	**5898**

As shown in Table III, DeblurGAN requires the least GPU memory at 4538 MB, while our method requires slightly higher GPU memory than DeblurGAN in GOPRO. For the VisDrone dataset, our network consumes the least GPU memory for the batch size of 16. The lightweight process can reduce the parameters of the model, thus contributing to the low-memory requirement.

TABLE IV: Average time of inferring images.

Method	GOPRO	VisDrone
	InferTime(s)	***InferTime(s)***
Deblur GAN	2.346	2.144
DMPHN	1.886	0.764
SIUN	0.684	0.357
LWRefineNet	**0.494**	**0.319**

As shown in Table IV, LWRefineNet is the fastest method in terms of the time of loading the network model and the inferences. The inference is run on GTX1650 4G GPU. The image size from GOPRO is 1280 × 768, while that from VisDrone is 256 × 256.

TABLE V: Quantitative numerical results on PSNR and SSIM.

Method	GOPRO		VisDrone	
	PSNR	***SSIM***	***PSNR***	***SSIM***
RefineNet	34.17826	0.894369	28.73991	0.854758
LWRefineNet	34.21445	0.906998	29.24461	0.860164
RCRefineNet	34.39430	0.903012	29.03971	0.858601
DFFNet	**34.63429**	**0.907881**	**29.40845**	**0.862474**

As shown in Table V, the LWRefineNet and RCRefineNet perform slightly better than RefineNet. DFFNet has the most significant numerical results.

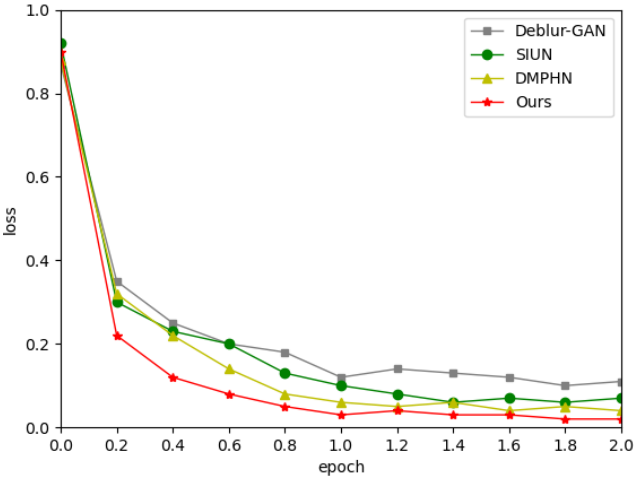
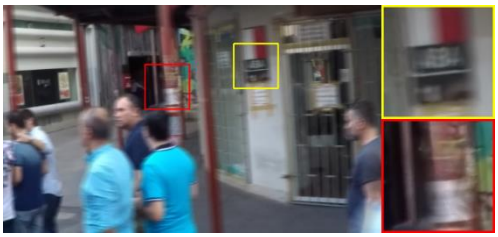


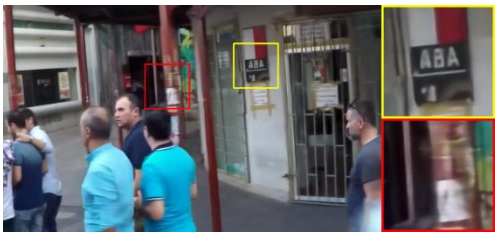
Fig. 5: Training loss of the four methods. Only the first two epochs are shown.

As shown in Fig. 5, our scale refinement loss function takes each sub-task as an independent component within a single unified task, allowing the training process to converge more rapidly and perform better than those of the other methods. The training losses of the other approaches decrease remarkably at the first round and consistently stay at 6% with a smooth trend in the following training courses. Our method, aided by the loss weight scheduling technique, exhibits a dramatic downward trend and remains at approximately 4%. The model accuracy improvements (approximately 10% to 21%) resulting from the multiple rounds of training for the four loss weight groups verify the good convergence and advantages of our method's training strategy.

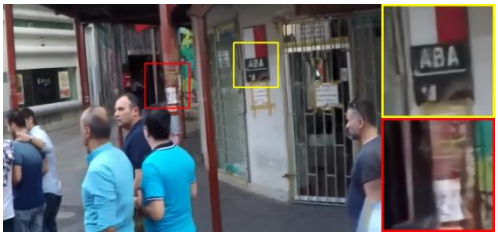


(a) Input image

(b) DeblurGan



(c) DMPHN

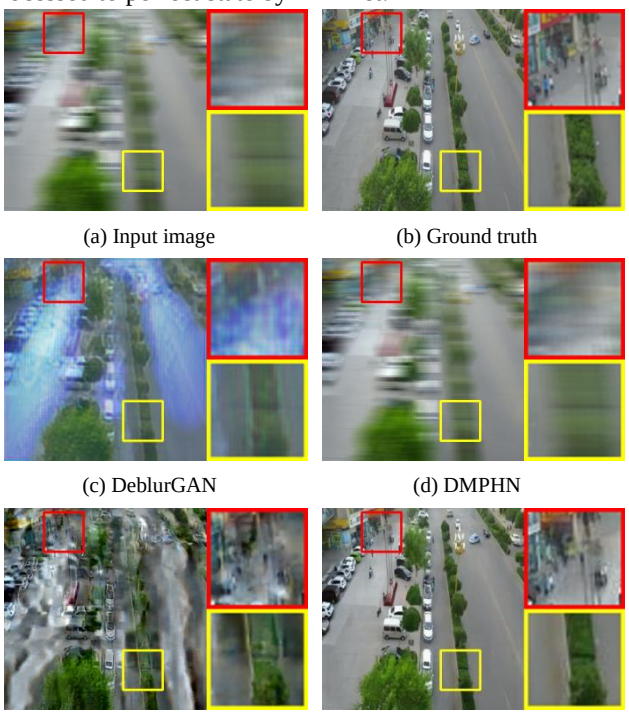


(d) SIUN

(e) Our method

Fig. 6: Visual effects of different methods. From top to bottom: blurred image, results of DeblurGAN, DMPHN, SIUN and ours. The left images are global deblur results, while local restoration details are shown on the right. Our results show clear object boundaries without artifacts.

The experimental results indicate that DFFNet can achieve considerable precision. Furthermore, DFFNet runs much faster than the other deblurring models, such as SIUN and DMPHN. Compared with DeblurGAN, the proposed DFFNet model performs well both in terms of speed and deblurring quality of images. Owing to the added lightweight process, the GPU memory occupation remains at a low level. Our method can also recover more details and achieve relatively high SSIM and PSNR values. Figs. 6 and 7 show that results of the other models whose images remain unstable and sometimes contain artifacts and color distortions, whereas the DFFNet performs image deblurring in a stable and sharp manner. For instance, the handwriting and flags in Fig.6 and the details in the streets in Fig. 7 are processed to perfect state by DFFNet.



(a) Input image (b) Ground truth

(c) DeblurGAN (d) DMPHN

(e) SIUN (f) Ours

Fig. 7: Results of comparative experiments. Our restored images show vivid colors and sharp details.

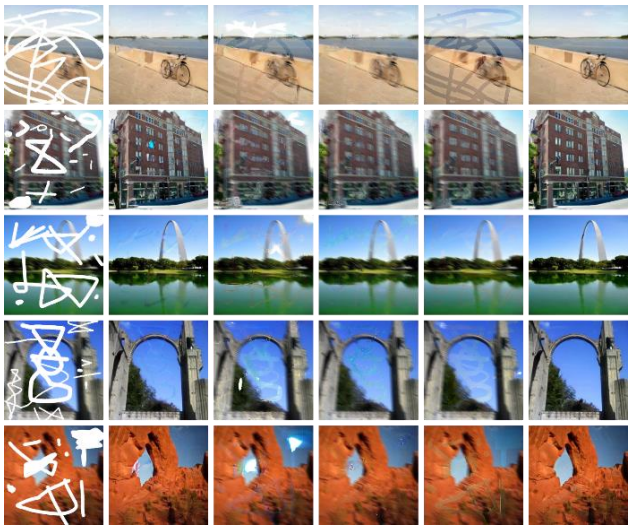
4.3.2 inpainting with state of the art

We conduct comparative experiments among CA, EC, LBAM, RN, and ours to verify the performance of our model.

Table 1: Testing results of deblur image datasets and PSNR and SSIM values.

Deblur Dataset	**VisDrone**		**Places2**	
	PSNR	**SSIM**	**PSNR**	**SSIM**
CA	34.36595	0.644143	33.17333	0.687329
EC	36.31839	0.839124	34.17416	0.855869
LBAM	34.15112	0.794887	34.39518	0.850359
RN	35.18645	0.815497	34.01265	0.865830
DFFNet	**36.86030**	**0.848545**	**34.52364**	**0.892832**

Our proposed DFFNet can achieve state-of-the-art performance compared with CA on VisDrone.



Input CA EC LBAM RN CRC

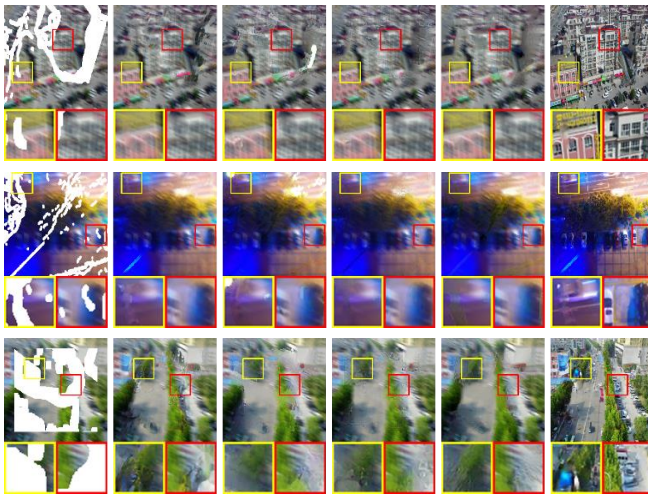
Figure 8: Results of inpainting on blur images on Places2.

Our PSNR and SSIM are much higher than EC, LBAM, and RN, suggesting the advantage of our method in handling Gaussian blurs and inpainting.

Table 2: Testing results of blur image datasets and PSNR and SSIM values.

Blur Dataset	VisDrone		Places2	
	PSNR	SSIM	PSNR	SSIM
CA	30.50928	0.625167	29.1809	0.615742
EC	30.3869	0.62006	29.16266	0.610526
LBAM	30.53208	0.625828	29.19904	0.609583
RN	29.9364	0.583832	29.03044	0.608648
DFFNet	**36.8603**	**0.848545**	**34.84132**	**0.893707**

Moreover, our method performs better than LBAM and EC and even much better than CA and RN in dealing with motion blurs and inpainting on Places2.



Input CA EC LBAM RN CRC

Figure 9: Results of inpainting on blur images on VisDrone.

The trends in Table 3 prove the superiority of the DFFNet framework based on PSNR and SSIM values.

Table 3: Average of L1 loss and L2 loss.

Deblur	Visdrone (%)		Places2 (%)	
	L1	L2	L1	L2
CA	19.30	6.73	18.30	8.4
EC	15.10	2.15	16.30	2.4
LBAM	24.00	8.91	22.00	7.9
RN	15.20	2.47	14.71	2.23
Ours	**12.45**	**2.03**	**9.05**	**1.53**

Figure 10 shows that our scale refinement loss function takes each subtask as an independent component within a single unified task, allowing training to converge more rapidly and perform better than other methods. The training losses of other approaches decrease remarkably in the first round and consistently stay at 6% with a smooth trend in following training courses.

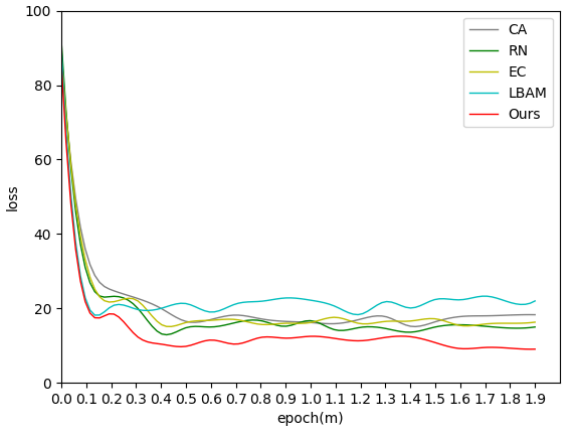


Figure 10: The loss function shows our training converges fast.

Our method, aided by loss weight scheduling technique, exhibits a dramatic downward trend and remains at approximately 4%. The model accuracy improvements (approximately 10% to 21%) resulting from multiple rounds of training for four loss weight groups verify the good convergence and advantages of our method's training strategy.

4.4. Ablation Experiments

The original benchmark is denoted as DFFNet. We add lightweight process to benchmark and denote it as DFFNet1. Then, we add remote residual connection to refinement path and residual connection to fusion unit, and denote it as DFFNet2. Finally, we combine lightweight and residual strategy to benchmark as DFFNet3.

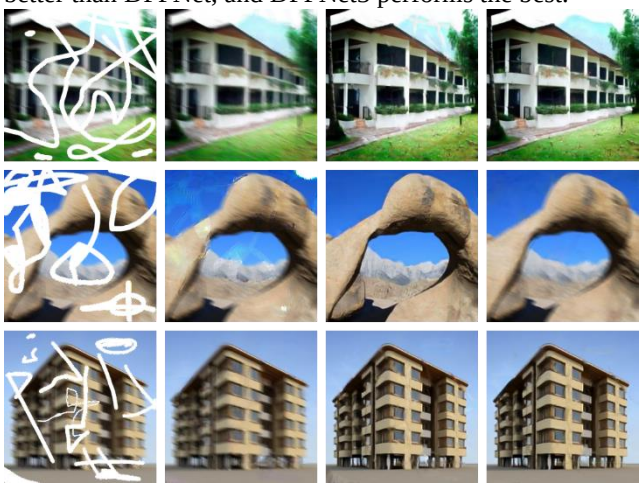
Table 4: Ablation results on PSNR and SSIM on deblur datasets.

Deblur Datasets	Visdrone		Places2	
	PSNR	SSIM	PSNR	SSIM
DFFNet1	36.66744	0.841927	34.45448	0.859129
DFFNet2	36.85104	**0.852030**	34.49627	0.865750
DFFNet3	**36.86030**	0.848545	**34.52364**	**0.892832**

Table 5: Ablation results on PSNR and SSIM on blur datasets.

Blur Datasets	Visdrone		Places2	
	PSNR	SSIM	PSNR	SSIM
DFFNet1	30.84918	0.718236	29.30308	0.731383
DFFNet2	36.85104	**0.852030**	34.49627	0.873377
DFFNet3	**36.86030**	0.848545	**34.84132**	**0.893707**

Tables 4 and 5 show that DFFNet1 and DFFNet2 perform better than DFFNet, and DFFNet3 performs the best.



Input DFFNet1 DFFNet2 DFFNet3

Figure 11: Visual effects of ablation experiments.

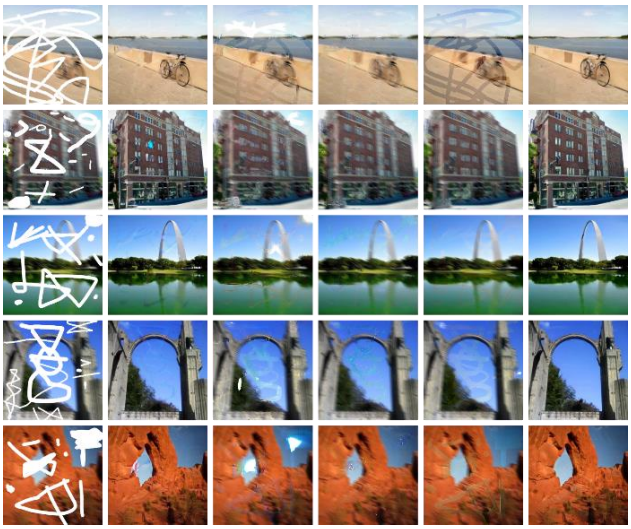
Figure 11 exhibits that our results show clear object boundaries without artifacts.

4.3.3 sharper inpainting with state of the art

We conduct comparative experiments by combining state-of-the-art deblurring and inpainting methods. The results of PSNR and SSIM are shown in Table 1.

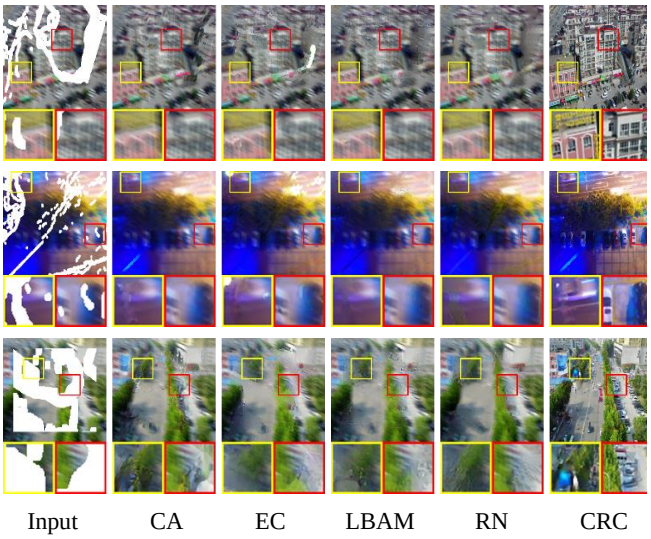
Table 1: Testing results of deblur image datasets and PSNR and SSIM values.

Deblur Dataset	VisDrone		Places2	
	PSNR	SSIM	PSNR	SSIM
deblurGAN+CA	30.50928	0.625167	29.1809	0.615742
deblurGAN+EC	30.3869	0.62006	29.16266	0.610526
deblurGAN+LBAM	30.53208	0.625828	29.19904	0.609583
deblurGAN+RN	29.9364	0.583832	29.03044	0.608648
DMPHN+CA	34.36595	0.644143	33.17333	0.687329
DMPHN+EC	36.31839	0.839124	34.17416	0.855869
DMPHN+LBAM	34.15112	0.794887	34.39518	0.850359
DMPHN+RN	35.18645	0.815497	34.01265	0.865830
DFFNet	**36.86030**	**0.848545**	**34.52364**	**0.892832**



Input CA EC LBAM RN CRC

Figure 8: Results of inpainting on blur images on Places2.



Input CA EC LBAM RN CRC

Figure 9: Results of inpainting on blur images on VisDrone.

Figure 10 shows the comparation of the training convergence tendency of the different inpainting methods.

Fig. 13. Comparison of 1-phase and 2-phase training performance on HFR-Net. The red circle indicates the switch from progressive to the retrogressive training phase. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Figure 13 shows that in the ablation experiments, the parameter α aims to control the proportion of the data flow

benefits the training effectively.

Table 4: Ablation results on PSNR and SSIM on deblur datasets.

Deblur Datasets	Visdrone		Places2	
	PSNR	SSIM	PSNR	SSIM
DFFNet 1	34.66744	0.841927	33.45448	0.859129
DFFNet 2	34.85104	0.848545	32.49627	0.865750
DFFNet 3	35.84918	0.818236	33.30308	0.841383
DFFNet 4	36.85304	0.832492	34.49627	0.873377
DFFNet 5	**36.86030**	**0.852030**	**34.84132**	**0.893707**

Without deblur feature of stage one is DFFNet1, decrease the context attention modules of stage 2 to 1 is DFFNet2, we define it as DFFNet 3 without using the progressive weighted training by parameter a. DDFNet 4 does not adopt the light weight and residual strategy. DDFNet5 does not remove any modules and strategies and outperforms among the ablation experiments.

Figure 11: Visual effects of ablation experiments.

5. Conclusion and Future Work

This paper proposed a multi-stage double feature fusion network FDDNet designed for deblurring and inpainting on the principle of explicit refinement and fusion of various restoration details. Based on this principle, the process began by deblurring the input to extract the sharp feature at first. Next, the network extracted and separated the refined feature patches in four multiscale extraction paths. Then each path predicts the holes with mask by multipath contextual attention modules which is aided by coherency algorithm. Finally, double features are fused and refined at the final stage and shaper predictions flows into the discriminator.

To generate the final prediction, the refined high-quality patches are fused with low quality images, and the fusion is performed using the proposed the progressive weighted training strategy. Additionally, for effectively image inpainting, multistage image inpainting contextual attention modules are introduced into the refinement fusion paths. Ablation and comparative experiments are conducted on three public datasets, and the results verifies the performance of DFFNet network and the effectiveness of all the proposed ideas.

The network shows that multi-stage workflows have a promising performance to process low quality images for multitask purpose. And the plug-and-play properties benefits the construction of network and flexibility of specific purposes. The progressive weighted training strategy is slower than the general training process due to the aided of ground truth images. However, the inference time of DFFNet is the same as the original benchmark. Thus, when considering performance has the highest priority, longer training time is not a significant limitation.

In the future, we will extract more features such as edge, structure or color and makes the synthesis inpainting more effectively and vividly. Several enhancements on attention modules could also be proposed to refine features and match patches more wisely and quickly, in addition, this network architecture could solve other restoration problems which remains to be explored such as dehazing and deraining.