A Progressive Fusion Generative Adversarial Network for Realistic and Consistent Video Super-Resolution

Peng Yi, Zhongyuan Wang, *Member, IEEE*, Kui Jiang, Junjun Jiang, *Member, IEEE*, Tao Lu, and Jiayi Ma

Abstract—How to effectively fuse temporal information from consecutive frames remains to be a non-trivial problem in video super-resolution (SR), since most existing fusion strategies (direct fusion, slow fusion, or 3D convolution) either fail to make full use of temporal information or cost too much calculation. To this end, we propose a novel progressive fusion network for video SR, in which frames are processed in a way of progressive separation and fusion for the thorough utilization of spatio-temporal information. We particularly incorporate multi-scale structure and hybrid convolutions into the network to capture a wide range of dependencies. We further propose a non-local operation to extract long-range spatio-temporal correlations directly, taking place of traditional motion estimation and motion compensation (ME&MC). This design relieves the complicated ME&MC algorithms, but enjoys better performance than various ME&MC schemes. Finally, we improve generative adversarial training for video SR to avoid temporal artifacts such as flickering and ghosting. In particular, we propose a frame variation loss with a single-sequence training method to generate more realistic and temporally consistent videos. Extensive experiments on public datasets show the superiority of our method over state-of-the-art methods in terms of performance and complexity. Our code is available at https://github.com/psychopa4/MSHPFNL

Index Terms—Convolutional neural network, video super-resolution, spatio-temporal correlation, progressive fusion, generative adversarial network

1 Introduction

Super-resolution (SR) refers to the problem of recovering a high-resolution (HR) output from a given low-resolution (LR) input. The SR technique has been a hot research issue in the past few decades, which has been applied to a lot of fields like image recognition [1], human face hallucination [2], remote sensing imagery [3], [4], etc. In the SR domain, various kinds of methods have been developed, e.g., interpolation-based methods [5], sparse coding-based methods [6] and learning-based methods [7]. With the rapid development of deep learning, learning-based SR has dominated the SR area in the last few years.

Despite the great success of single image super-resolution (SISR) with the help of deep learning, there is still a big gap for video SR: how to effectively fuse spatio-temporal information between frames, how to deal with motions among frames, and how to reconstruct both realistic and temporally consistent videos.

As for the first issue, a lot of video SR methods [8], [9], [10] naively learn the architecture from SISR without significant improvements. For single image super-resolution, as the input consists of only one single image, it mainly focuses on extracting spatial information within one image. For video SR, where the input is composed of a certain number of LR frames, it puts more emphasis on exploiting the inter-frame temporal information. Thus, a deep learning framework that can make full use of spatio-temporal correlations but is computationally efficient is the key to video SR.

In general, most current video SR methods either adopt a time-based way or space-based way for utilizing temporal information. Time-based methods [11], [12], [14], [15], [16], [17] consider frames as time-series data, and send frames through the network in a specific sequence. Nevertheless, they have to consider how to retain long-term dependencies effectively. Theoretically, this kind of approaches may require more time than space-based approaches do, as it has to process frames sequentially and cannot handle multiple frames in parallel. As illustrated in Fig. 1, two kinds of time-based methods [11], [12] both treat LR frames in sequence.

Space-based methods [8], [9], [18] take multiple frames as supplementary materials to help reconstruct the reference frame. Compared to time-based methods, this kind of methods is able to retain full inter-frame temporal correlations and enjoy the advantages of parallel computing. Most existing space-based approaches employ direct fusion [9], [19], slow

- *Peng Yi, Zhongyuan Wang, and Kui Jiang are with the National Engineering Research Center for Multimedia Software, School of Computer Science, Wuhan University, Wuhan 430072, China. E-mail: {yipeng, kuijiang}@whu.edu.cn, wzy_hope@163.com.*
- *Junjun Jiang is with the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China, and also with the Peng Cheng Laboratory, Shenzhen 518066, China. E-mail: junjun0595@163.com.*
- *Tao Lu is with the School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430205, China. E-mail: lutxyl@gmail.com.*
- *Jiayi Ma is with the Electronic Information School, Wuhan University, Wuhan 430072, China. E-mail: jyma2010@gmail.com.*

Manuscript received 9 Nov. 2019; revised 31 Aug. 2020; accepted 28 Nov. 2020. Date of publication 3 Dec. 2020; date of current version 1 Apr. 2022.
(Corresponding authors: Zhongyuan Wang.)
Recommended for acceptance by G. Shakhnarovish.
Digital Object Identifier no. 10.1109/TPAMI.2020.3042298

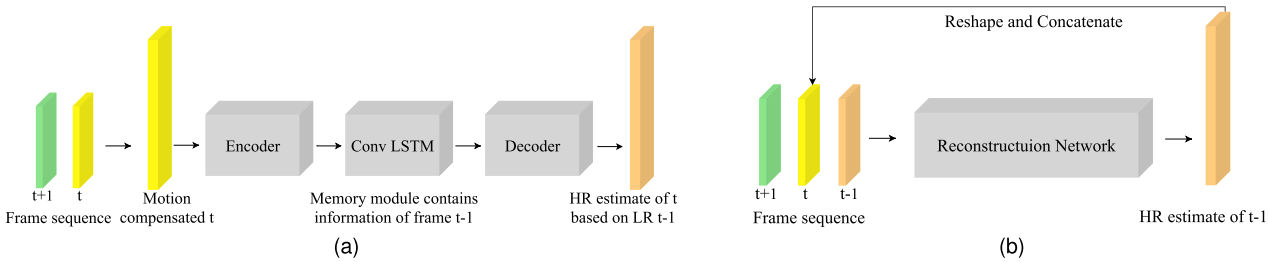


Fig. 1. Different merging schemes for time-based video SR. (a) Architecture of DRVSR [11]. (b) Architecture of FRVSR [12].

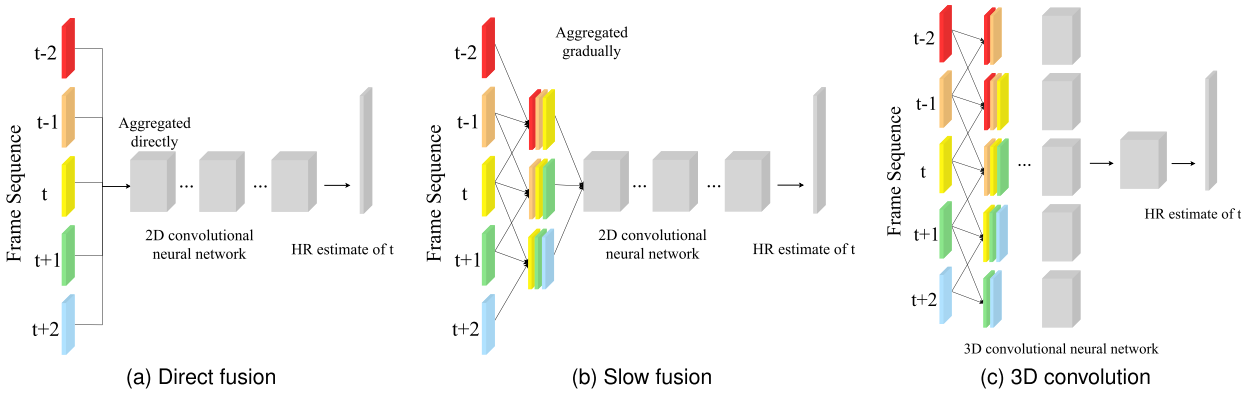


Fig. 2. Different merging schemes [9] for space-based video SR, when adopting five frames as input. (a) Direct fusion strategy fuses multiple frames into one part in the beginning. (b) Slow fusion strategy fuses frames into smaller groups gradually and into one piece. (c) A 3D convolution not only convolutes frames in the space dimension, but also in the time dimension.

fusion [8], [9] or 3D convolution [20], [21] for temporal information fusion [9], which are shown in Fig. 2. However, direct fusion and slow fusion naively treat video SR as SISR and have to stack multiple frames as a whole at the beginning, without explicitly considering the temporal structure in terms of frame, while 3D convolution involves huge computational costs. Unlike them, we adopt a multi-channel progressive fusion, where video frames are assigned to separate channels in order and then merged by a series of progressive fusion residual blocks. This way, both the fusion of spatio-temporal correlation and the independence of individual frames are accounted for.

Different from still images, video frames contain moving scenes and objects, and large-size objects within frames and large motion displacement across frames call for a large receptive field. Dilated convolutions [22], [23] are a common way to expand the receptive field without extra parameter requirement. However, simply enlarging the dilation rate may cause the network to fail to extract local features from neighbouring pixels, thus compromising the overall performance. The multi-scale structure has been applied to areas like classification [24], object detection [25] and image deraining [26], *etc*. Although it is usually used to extract hierarchical features of an image, we alternatively use it to expand the receptive field for capturing large-scale objects or large inter-frame displacement. For instance, by down-sampling image features by a factor of 2, a convolution has access to pixels twice as far as before.

For the second issue, most traditional video SR methods [27], [28], [29], [30] conduct pixel-level motion and blur kernel estimation based on image priors, trying to solve a complex optimization problem [10]. With recent development of deep learning, a lot of video SR methods [8], [9], [10], [11], [12] based on convolutional neural networks have emerged. These methods attempt to conduct explicit motion compensation on the input frames to eliminate the motions among frames as a preprocessing step. Although these alignment methods are intended for increasing the temporal coherence, they have two main disadvantages: 1) they introduce extra parameters, computational cost, as well as training difficulty, and 2) inaccurate motion estimation and motion compensation (ME&MC) may corrupt original frames and lower the performance of SR [31]. Besides, by stacking of convolutional layers, the large receptive fields of deep CNNs already make it possible to capture the motions from a wide range, thus, deep CNNs do not need the help of ME&MC necessarily. We have conducted experiments to find that these ME&MC [9], [11] methods contribute little (about 0.01-0.04 dB) to the video SR in Section 4.3.

Recently, inspired by image denoising [32], non-local neural networks [13] have been proposed by Wang *et al*. for capturing long-range dependencies. From our perspective, although this paradigm does not aim to eliminate the motions, it also intends to increase the temporal coherence among frames, which means that it enjoys the potential to replace the complex ME&MC in video SR. As illustrated in Fig. 3, non-local operation aims at computing the correlations between all possible pixels within and across frames, while ME&MC intends to compensate other frames to the reference frame as close as possible.

Few video SR methods [19] have tried to solve the third issue. Recent most SR methods [18], [33], [34], [35], [36], [37], [38] concentrate on increasing the numerical values, mostly PSNR and SSIM of the reconstructed images. However, these

Fig. 3. Differences of non-local operation and ME&MC. Non-local operation tries to obtain the response at position x_i by computing the weighted average of relationships of all possible positions x_j [13]. ME&MC tries to compensate neighboring frames to the reference frame. More specific explanations can be found in Section 3.2.

methods are prone to generate overly smooth and blurry images lacking realistic details. SRGAN [39] and ESRGAN [40] have introduced generative adversarial network (GAN) into SISR to generate photo-realistic SR images. However, due to the instability of generative adversarial learning, GANs have seldom been applied to video SR yet. We have conducted experiments using GAN for video SR, as shown in Fig. 14, and we have found that although the reconstructed video frames seem to contain more high-frequency details, they are inconsistent in temporal (like ghosting and flickering). To produce temporally consistent video frames, we additionally propose a frame variation loss (FVL) function together with a single-sequence training method especially for GAN's training. By doing so, our SR frames not only contain more realistic details, but also maintain more consistent in temporal.

In summary, our major contributions are highlighted as follows:

1) We propose a novel progressive fusion network by incorporating a series of progressive fusion residual blocks (PFRBs), which extracts and fuses spatio-temporal information from multiple frames progressively in a multi-channel structure, thus enabling our model more effective and efficient than preceding models. Meanwhile, we devise a cross-channel parameter sharing (CCPS) scheme to reduce the parameter number of a PFRB in a great deal while retaining its performance.
2) We incorporate hybrid convolutions (both normal and dilated convolutions) and multi-scale structure into video SR so that our model has access to pixels from a larger receptive field, thus better capturing long-range spatio-temporal correlations brought by motions among frames.
3) We propose the non-local residual block (NLRB) to directly capture long-range spatio-temporal correlations without requiring extra ME&MC algorithms.
4) We introduce GAN into video SR and accordingly propose a frame variation loss function together with a single-sequence training (SST) method to generate more spatially realistic and temporally consistent videos.

This paper is improved on our previous conference work [36] in several aspects. Above of all, we re-examine the challenges of video SR unlike from SISR and summarize them into three major issues. As discussed above, one is related to the overall spatio-temporal fusion framework, and the other two are related to the methods capturing the inter-frame motion and maintaining temporal consistency, respectively. Our previous work actually tried to address the former two issues, while in this work we further incorporate the multi-scale structure into our network to expand the receptive field accounting for both the spatial coverage and temporal displacement required for consecutive frames. More importantly, we introduce an additional GAN based training method specific for handling the third issue in this work. In particular, we have established a frame variation loss and a single-sequence training method, which enables our network to reconstruct both spatially realistic and temporally consistent videos. In addition, we have explored and discussed the influence of input size on training in video SR, by evaluating the models trained with different sizes of input. Also, we have rebuilt another four latest video SR methods [15], [16], [17], [41], two GAN-based SISR methods [39], [40] and their video SR versions. We conduct a comprehensive comparison of different methods on performance, number of parameters, testing time cost, training time cost, number of GPUs required for training, as well as visual results. Lastly, we have conducted additional experiments on a larger public dataset Vimeo-90K [31] under a new down-sampling scheme to prove the generalization ability of our model. Through the improvements and added exploration, the performance of our method has been significantly boosted compared with the previous work, which enjoys an about 0.45 dB gain in average experimentally. We compare our method with dozens of state-of-the-art methods, from which it is proven that our method not only yields the best performance, but also requires a relatively small number of parameters, time cost and less training resources.

The remaining of this paper is organized as follows. In Section 2, we introduce the development for both SISR and video SR. In Section 3, we elaborate the design of our network. In Section 4, we conduct various ablation studies to prove the effectiveness and efficiency of our model. In Section 5, we compare our network with dozens of state-of-the-art methods. In Section 6, the paper is concluded.

2 Related Work

2.1 Single Image Super-Resolution

As is universally acknowledged, SISR is the basis in the SR area, from which video SR has benefited a lot. Here, we discuss recent studies in SISR and their influences on video SR.

Since Dong *et al.* proposed a super-resolution convolutional neural network (SRCNN) [7], [42], deep learning has appealed to an enormous number of researchers in SISR, and a large quantity of CNN-based SISR methods [33], [43],

[44] have been proposed. Kim *et al.* presented a very deep convolutional network (VDSR) [33], where a residual learning scheme is introduced to SISR. This network tries to learn the residual information between LR and HR images, thus alleviating the burden of the network and enabling the network especially deep. Shi *et al.* came up with an efficient sub-pixel convolutional neural network (ESPCN) [43], from which a sub-pixel convolutional layer is used to replace the traditional transposed convolutional layer. The sub-pixel convolutional layer has been adopted by a lot of methods [39], [45], [46], [47]. Lai *et al.* proposed a deep laplacian pyramid network (LapSRN) [48] that adopts the multi-scale training strategy to train a single model for handling multiple up-sampling scales [34]. Zhang *et al.* put forward a residual dense block (RDB) [49], which incorporates local and global skip connection and concatenation to extract features. Further, a very deep residual channel attention network (RCAN) [35] was also proposed to utilize the channel attention mechanism to boost the performance. Zhang *et al.* explored the dilated convolutions in SR area by proposing dilated convolutions for SISR (DCSR) [50]. Ledig *et al.* proposed a super-resolution generative adversarial network (SRGAN) [39] to generate photo-realistic images, from which an enhanced version (ESRGAN) [40] was devised by Wang *et al.* to synthesize more realistic details. Zhang *et al.* put forward a RankSRGAN [51] to optimize the generator in terms of perceptual metrics. Shaham *et al.* [52] designed a pyramid of multi-scale patch-wise generators and discriminators, which processes images at different scales according to their levels.

In all, influenced by these SISR methods, architectures for video SR have been improved and developed.

2.2 Video Super-Resolution

Based on how the input frames are processed, video SR methods can be categorized into two ways: time-based way and space-based way. Two kinds of time-based methods [11], [12] and three kinds of space-based fusion strategies [9], [18], [53] are illustrated in Figs. 1 and 2. In general, time-based methods treat LR frames in a certain order, usually one by one each time, and they have to consider how to retain and transmit information from different time stages. Different from time-based methods, space-based methods try to handle LR frames at the same time, and they focus on how to fuse information from different frames at one time.

As shown in Fig. 1a, Tao *et al.* followed the time-based way and proposed a detail-revealing deep video super-resolution network (DRVSR) [11], which adopts a convolutional long short-term memory (ConvLSTM) [54] module, through which the inter-frame temporal information can be reserved. Based on spatial transformer motion compensation (STMC) [9], DRVSR also proposes a novel sub-pixel motion compensation (SPMC) layer, warping and projecting LR frames onto HR image space in the meantime. Nevertheless, the performance of SPMC layer is limited [36], which also requires more GPU memory and calculation cost [15], [16]. Based on DRVSR, Wang *et al.* presented a multi-memory convolutional neural network for video super-resolution (MMCNN) [15], which incorporates multiple ConvLSTMs to enhance the memory ability. Recently, frame-recurrent video super-resolution (FRVSR) [12] was put forward by Sajjadi *et al.* As illustrated in Fig. 1b, FRVSR first sends one LR frame through a network to obtain a super-resolved HR output. Then, this HR output is reshaped and concatenated to the next LR frame, flowing through the network again to produce the corresponding HR estimate. By this way, as frames go through the network, their outputs are sent back to assist to reconstruct subsequent frames. However, FRVSR can only utilize previous frames to help reconstruct the current LR frame, neglecting the potential of next LR frames. More recently, based on [55], Haris *et al.* came up with a recurrent back-projection network for video super-resolution (RBPN) [17], in which each context frame is treated as a separate source of information and these sources are sent to the network iteratively. Whereas the good performance of the RBPN, it requires prohibitive GPU numbers (8 NVIDIA TITAN X GPUs) and memory for training due to the frequent up-sampling and down-sampling layers. Generally speaking, how frames are treated by time-based methods corresponds to the nature of time-series data, whereas, as has been discussed before, they are disadvantageous at parallel computing.

Space-based methods [8], [9], [10], [18], [41], [53] try to merge temporal information in a parallel manner, and three main fusion strategies are illustrated in Fig. 2. Direct fusion and slow fusion share a similar idea, except that the former fuses frames into one part directly, while the latter does the same thing gradually. A 3D CNN conducts convolution in both space and time dimensions, extracting intra-frame spatial correlations and inter-frame temporal correlations simultaneously. Kappeler *et al.* proposed a video super-resolution with convolutional neural network (VSRnet) [8] to follow direct fusion. Jo *et al.* presented a deep video super-resolution network using dynamic up-sampling filters (DUFVSR) [18] to build a 3D CNN, while it involves huge calculation and time cost. Caballero *et al.* came up with a real-time video super-resolution with spatio-temporal network and motion compensation (VESPCN) [9], and this work has explored these three strategies for video SR. However, the naive network design has limited the performances of these methods [8], [9]. Liu *et al.* proposed a robust video super-resolution with learned temporal dynamics (RVSR-LTD) [10], in which LR frames are divided into different inference branches, while these inference branches have to deal with redundant information unnecessarily. Wang *et al.* put forward a video restoration with enhanced deformable convolutional network (EDVR) [41], where an alignment module known as Pyramid, Cascading and Deformable convolutions (PCD) is proposed to align neighboring frames to the reference frame at the feature level. Through a sophisticated framework as well as large batch size (32) and input size (64×64) in training, EDVR has achieved a remarkable performance, but also with prohibitive resources required (8 NVIDIA Titan Xp GPUs).

Our proposed progressive fusion strategy also follows the space-based way, thus, we have conducted experiments to compare the proposed progressive fusion strategy against these three strategies demonstrated in Fig. 2. Various experiments indicate that our proposed network is not only more effective but also more efficient than other fusion strategies, and the details are elaborated in Section 4.

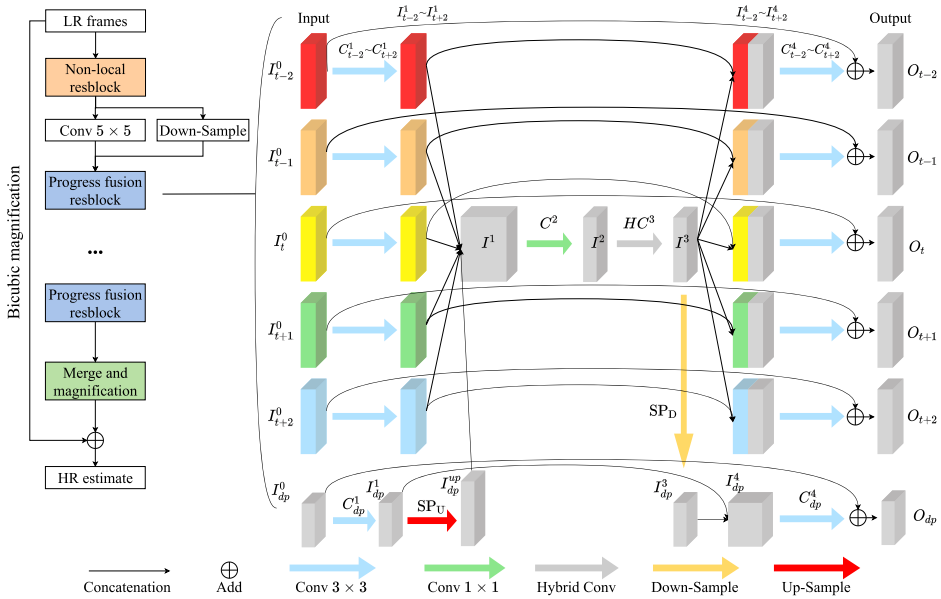


Fig. 4. The architecture of our whole network (left) and the structure of one PFRB (right). Note that the input number of the residual blocks can be arbitrary, and we show the case taking five frames as input. The "⊕" refers to element-wise adding. Black, blue, green, gray, yellow and red arrows denote concatenation, normal convolution with 3×3 kernel, normal convolution with 1×1 kernel, hybrid convolution, down-sample, and up-sample function, respectively.

3 Our Method

3.1 Progressive Fusion Network

We first present the design of our whole network. As illustrated in Fig. 4, we adopt a non-local block to enforce the coherence of input LR frames in the beginning. Then, after one 5×5 convolution layer and a down-sampling module, we replenish a series of PFRBs to the network, which are supposed to make full extraction of both intra-frame spatial correlations and inter-frame temporal correlations among multiple LR frames. At the end, we merge the information from all channels in PFRB and enlarge it to obtain one residual HR image, which is added to the bicubically magnified image to produce the HR estimate as in [33]. Merge and magnification module consists of 4 layers: one 3×3 convolutional layer with 48 filters and one 3×3 convolutional layer with 12 filters, each of which is followed by a sub-pixel magnification layer [43] for $2\times$ upscaling. The whole processing procedure is quite straightforward, nevertheless, we have designed two sophisticated residual blocks for better utilization of spatio-temporal information.

We show the structure of a PFRB in the right of Fig. 4, when taking five frames as input. Basically, we follow a multi-channel design methodology, because the input contains multiple LR frames. The input includes 5 feature maps $\{I^1_{t-2}, \ldots, I^1_{t+2}\}$ convoluted from 5 consecutive frames and 1 feature map I^0_{dp} at a smaller size, which is down-sampled from all 5 concatenated frames after the NLRB. We choose 2 as the factor of down-sampling, and the same below. We first conduct 3×3 convolutional layers with same certain number (N) of filters, which can be described as

$$I^1_t = C^1_t(I^0_t), \tag{1}$$

$$I^1_{dp} = C^1_{dp}(I^0_{dp}), \tag{2}$$

where t denotes the index of temporal dimension, I^0_t represents the input frames, and $C^1_t(\cdot)$ is the first depth of the convolutional layers. I^1_t indicates the feature maps extracted, which are supposed to contain rather self-independent features from each input frame. I^0_{dp} and I^1_{dp} represent the corresponding feature maps at a smaller scale, which ensure the increment of receptive field. C^1_{dp} is a convolution uninfluenced by the size of the input feature map. Small-size feature map I^1_{dp} is further up-sampled to I^{up}_{dp} with the original size

$$I^{up}_{dp} = \mathrm{SP}_U(I^1_{dp}), \tag{3}$$

where $\mathrm{SP}_U(\cdot)$ is an up-sampling function composed of a 1×1 convolution and a sub-pixel convolution layer.

Later, feature maps $\{I^1_{t-2}, \ldots, I^1_{t+2}, I^{up}_{dp}\}$ are concatenated and merged into one part, which contains information from all input frames at different scales, and the depth of this merged feature maps is $(5+1) \times N$ when taking five frames as input. Alternatively, this aggregated deep feature map contains a large deal of multi-scale temporally correlated information. Naturally, we use one convolutional layer to further leverage the temporal information from this feature map. Because this feature map is rather deep, we set the kernel size as 1×1 to avoid involving too many parameters. Considering the fact that the temporal correlations between two frames are negatively correlated with the temporal distance, we set the filter number as N to distillate this deep feature map to a more concise one, intending to extract features that are most temporally correlated to the center frame. This process can be described as

$$I^2 = C^2(I^1) = C^2(\{I^1_{t-2}, \ldots, I^1_{t+2}, I^{up}_{dp}\}), \tag{4}$$

where I^1 denotes the aggregated feature map, $C^2(\cdot)$ represents the second depth of the convolutional layer with 1×1 kernel, and I^2 is the distillated feature map.

Later, we incorporate the dilated convolution into this module to utilize pixels from a larger receptive field. Instead of using a dilated convolution directly, we design a hybrid convolution that benefits from both normal convolutions and dilated convolutions. Specifically, a hybrid convolutional layer with N filters consists of 3 layers, where it first conducts a 3×3 normal convolutional layer with $N/2$ filters and a 3×3 dilated convolutional layer (dilation rate is 2) with $N/2$ filters to extract both local features and wide-range features in the meantime. This way, a hybrid convolutional layer is able to handle pixels in a receptive field as large as a 5×5 normal convolutional layer can handle, but with fewer number of parameters. The local features and wide-range features are then concatenated together and processed by a 1×1 normal convolutional layer with N filters for better information fusion. In all, this process can be described as

$$I^3 = HC^3(I^2) = C^3_{1\times1}(\{C^3_{3\times3}(I^2), DC^3_{3\times3}(I^2)\}), \tag{5}$$

where I^3 denotes the output feature map, $HC^3(\cdot)$ represents the hybrid convolutional layer, $C^3_{3\times3}(\cdot)$ and $DC^3_{3\times3}(\cdot)$ are a normal convolutional layer and a dilated convolutional layer both in 3×3 kernel, and $C^3_{1\times1}(\cdot)$ denotes a normal convolutional layer with 1×1 kernel.

After that, we down-sample feature map I^3 to I^3_{dp}

$$I^3_{dp} = \mathrm{SP}_D(I^3), \tag{6}$$

where $\mathrm{SP}_D(\cdot)$ is a down-sampling function composed of a 1×1 convolution and a reshape operation. Then, I^3 and I^3_{dp} are concatenated to all the previous feature maps $\{I^1_{t-2}, \ldots, I^1_{t+2}\}$ and I^1_{dp} respectively, which become $\{I^4_{t-2}, \ldots, I^4_{t+2}\}$ and I^4_{dp}. Thus, feature maps of all channels contain two kinds of information now: self-independent spatial information and fully mixed temporal information. Besides, the spatio-temporal information are extracted from longer range, which benefits much from the multi-scale structure and a hybrid convolution. It is easily observed that the depth of these feature maps is $2 \times N$, and we further adopt 3×3 convolutional layers to extract both spatial and temporal information from them, formulated as

$$O_t = C^4_t(I^4_t) + I^0_t = C^4_t(\{I^3, I^1_t\}) + I^0_t, \tag{7}$$

$$O_{dp} = C^4_{dp}(I^4_{dp}) + I^0_{dp} = C^4_{dp}(\{I^3_{dp}, I^1_{dp}\}) + I^0_{dp}, \tag{8}$$

where I^4_t denotes the merged feature maps $\{I^3, I^1_t\}$, $C^4_t(\cdot)$ represents the fourth depth of the convolutional layers, and O_t is the corresponding output. I^4_{dp} indicates the merged down-sampled feature maps $\{I^3_{dp}, I^1_{dp}\}$, $C^4_{dp}(\cdot)$ is a convolution, and O_{dp} means the output in a smaller scale. We use "$+I^0_t$" and "$+I^0_{dp}$" to represent residual learning [56], which means that the output and the input are in the same size. Thus, the fourth depth of convolutional layers is equipped with N filters, used for obtaining N-depth outputs with more concise and efficient information.

Overall, we design a sophisticated residual block that keeps the number and size of the input frames unchanged, where both intra-frame spatial correlations and inter-frame temporal correlations are fully exploited progressively. By involving the multi-scale structure and dilated convolutions, the proposed PFRBs have a larger receptive field to capture longer-range dependencies. Although the parameter number of one PFRB grows linearly with the number of input frames, we further use a parameter sharing scheme to reduce the parameters with barely no loss in performance. Different from the schemes in [57] and [58], where they share the parameters across different residual blocks, we perform parameter sharing across channels within one residual block, which will make the parameters of one PFRB nearly uninfluenced by the number of input frames. Specifically, the weights of convolutions at the same layer ($\{C^1_{t-2}, \ldots, C^1_{t+2}, C^1_{dp}\}$ and $\{C^4_{t-2}, \ldots, C^4_{t+2}, C^4_{dp}\}$) are shared within this layer. We also conduct comparisons between our network and direct fusion, slow fusion and 3D convolution strategies with different parameter sharing schemes, which are shown in Section 4.4.

3.2 Non-Local Residual Block

As shown in Fig. 3, the non-local operation calculates the correlations between all pixels among frames. Intuitively, inspired by [13], [59], we introduce a kind of non-local operation to capture long-range dependencies in one module. Mathematically, the non-local operation is described as follows:

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j) g(x_j), \tag{9}$$

where x represents the input data (image, video, text, etc), y denotes the output, which should be the same size as x. i is the index of an output position, and j is the index of all possible positions. The function $f(\cdot)$ calculates a scalar that represents a kind of relationship between two inputs, while $g(\cdot)$ indicates a representation of the input. $C(x)$ is used for normalization.

According to Equation (9), it can be seen that this non-local operation tries to obtain a representation at one position by computing the correlations between it and all possible positions. This is the exact reason we choose it to replace traditional ME&MC. Wang *et al.* [13] provided some choices for function $f(\cdot)$ like Gaussian function, embedded Gaussian function and dot product function, *etc*. To avoid involving too many extra parameters, we only consider the Gaussian function $f(x_i, x_j) = e^{x_i^T x_j}$, where $x_i^T x_j$ represents the dot-product similarity, and $C(x) = \sum_{\forall j} f(x_i, x_j)$ is used for normalization.

As demonstrated in Fig. 5, the output of an NLRB is defined as $z_i = W_z y_i + x_i$, where W_z is implemented by 1×1 convolution, and "$+x_i$" is referred to as the residual learning [56]. We first reshape $\boldsymbol{X}$ to $\boldsymbol{X_1}$, transforming the temporal dimension into the channel dimension. In other words, the temporal correlations are not calculated directly, but captured through the channel correlations. If without doing so, the shape of feature map $\boldsymbol{F}$ would be $THW \times THW$, which is the same as [13]. However, non-local block is originally proposed for tasks like classification and recognition, where the input size H and W have been down-scaled quite small and fixed, e.g., 28, 14 or even 7. Nevertheless, for generation task like SR, the input size can be arbitrary and much bigger, e.g., 480×270, where $\boldsymbol{F}$ could be too big to compute and store. Thus, we have to redesign NLRB to make it suitable for video SR. Further, based on [43], we reshape $\boldsymbol{X_1}$ to $\boldsymbol{X_2}$, transforming part of the spatial dimension into the channel dimension,

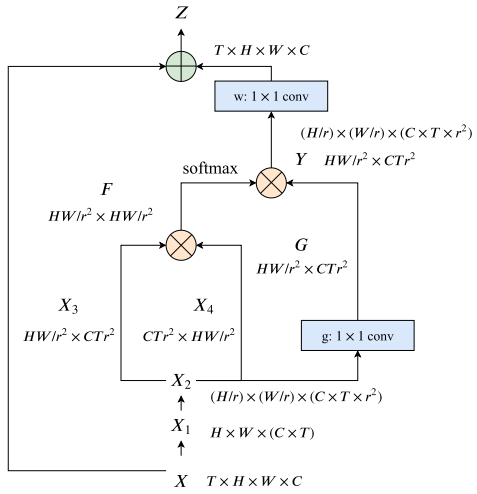


Fig. 5. Structure of one NLRB. We show the feature maps as their shapes like $T \times H \times W \times C$, and they are reshaped if noted. The "$\otimes$" represents matrix multiplication and "$\oplus$" represents element-wise add.

with r denoting the down-scaling factor. By that means, the shape of feature map $\boldsymbol{F}$ becomes $HW/r^2 \times HW/r^2$, which is unrelated to number of frames T any more and can be adjusted through r. By the two improvements, our model is able to generate full HD video frames under $4\times$ SR. For $4\times$ video SR, we choose $r = 2$ to make our network capable of reconstructing full HD 1920×1080 video frames from 480×270 LR input frames. As an NLRB does not change the input shape, it can be inserted into the existing models conveniently. We show the superiority of the NLRB against ME&MC in Section 4.3.

3.3 Generative Adversarial Training for Video Super-Resolution

Numerous video SR methods have been proposed in recent years, whereas they focus on improving the SR performance in terms of PSNR values rather than the subjective visual qualities [12], [15], [16], [17], [18], [36], [41]. Several methods [39], [40], [51], [60] have been proposed to improve the visual performance in SISR, which are capable of recovering high-frequency details from LR images. We now give a description of GAN learning structure in SR, as illustrated in Fig. 6. The generator aims to magnify the LR input, while the discriminator tries to distinguish between fake SR and real HR images. We learn relativistic average discriminator (RaD) from [40], [61], formulated as

$$D_{Ra}(I_i^{HR}, I^{SR}) = \sigma(C(I_i^{HR}) - \frac{1}{N}\sum_{i=1}^{N} C(I_i^{SR})), \tag{10}$$

where I^{HR} denotes the real HR image, I^{SR} is the fake SR image, N means the size of a mini-batch, σ represents the Sigmoid function, and $C(\cdot)$ indicates non-transformed discriminator output [61].

Then, the adversarial loss for discriminator ($\mathcal{L}_D^{Ra}$) and generator ($\mathcal{L}_G^{Ra}$) can be respectively defined as

$$\mathcal{L}_D^{Ra} = -\frac{1}{N}\sum_{i=1}^{N}[\log(D_{Ra}(I_i^{HR}, I^{SR})) + \log(1 - D_{Ra}(I_i^{SR}, I^{HR}))], \tag{11}$$

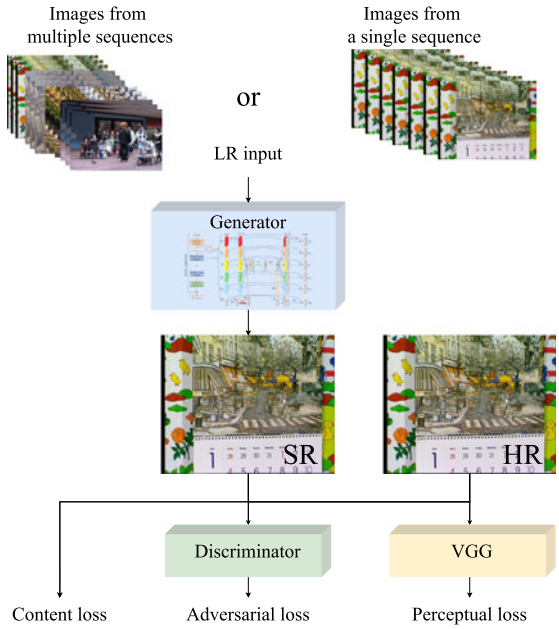


Fig. 6. A progressive fusion GAN framework for video SR, where the generator is implemented based on the progressive fusion model in Fig. 4 and the discriminator is composed of ESRGAN [40].

$$\mathcal{L}_G^{Ra} = -\frac{1}{N}\sum_{i=1}^{N}[\log(1 - D_{Ra}(I_i^{HR}, I^{SR})) + \log(D_{Ra}(I_i^{SR}, I^{HR}))], \tag{12}$$

where $I^{SR} = G(I^{LR})$, I^{LR} represents the LR input, and $G(\cdot)$ denotes the function of generator.

The perceptual loss has been proposed by [60] and developed in SISR [39], [40], [51], which helps guide the generator to recover visually more pleasant SR images. We follow perceptual loss $\mathcal{L}_P$ from [40]:

$$\mathcal{L}_P = \frac{1}{N}\sum_{i=1}^{N}\|\phi(I_i^{SR}) - \phi(I_i^{HR})\|_1, \tag{13}$$

where $\phi(\cdot)$ denotes the pretrained 4th convolution before the 5th maxpooling layer of VGG19 network [62]. Through perceptual loss, the differences of activated features between fake SR and real HR are minimized.

The content loss has always been the basic loss function for SISR and video SR, which can be defined as

$$\mathcal{L}_C = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(I_i^{SR} - I_i^{HR})^2 + \varepsilon^2}, \tag{14}$$

where ε is a small constant (usually set to 10^{-3}). Charbonnier loss [34] is used here, while L1 and L2 loss are alternatives. Content loss ensures that the reconstructed SR images and HR images are as close as possible at pixel level.

However, as demonstrated in Fig. 14, after applying SRGAN [39] and ESRGAN [40] to video SR, albeit the reconstructed video frames show to be sharper, they meanwhile suffer from various annoying artifacts, such as noise, unnaturalness, ghosting, flickering, and aliasing. We further adopt direct fusion strategy (DFS) shown in Fig. 2a to SRGAN and ESRGAN, and train their corresponding video

SR counterparts VSRGAN and VESRGAN. The results of VSRGAN and VESRGAN contain fewer artifacts, but they are still inconsistent in temporal. As has been discussed above, traditional content loss only minimizes the distance between SR images and HR images, which is suitable for SISR. Nevertheless, in video SR, it cannot guarantee that the variations between SR frames are close to the variations between HR frames. Without the GAN, the video SR results are overly smooth and blurry, and it is hard to observe the sudden changes between SR frames normally. However, after introducing GAN into video SR, as more high-frequency details are generated, the inconsistent variations between SR frames become obvious and distracting.

From these observations, we have realized that generating both realistic and temporally consistent videos requires more than traditional content loss and the training method by SISR. Therefore, we propose the following frame variation loss function:

$$\mathcal{L}_{FV} = \frac{1}{T-1}\sum_{i=1}^{T-1}\sqrt{\left|(I_{i+1}^{HR} - I_i^{HR})^2 - (I_{i+1}^{SR} - I_i^{SR})^2\right| + \varepsilon^2}, \tag{15}$$

where the input LR frames are from one single video sequence, thus the corresponding SR frames are consecutive, and T denotes the number of frames. In practice, we use FVL to constrain the generator, trying to restrict the changes between SR frames to be consistent with that between HR frames. Equation (15) can be further rewritten as

$$\mathcal{L}_{FV} = \frac{1}{T-1}\sum_{i=1}^{T-1}\sqrt{\begin{aligned}&|(I_{i+1}^{2HR} - I_{i+1}^{2SR}) + (I_i^{2HR} - I_i^{2SR}) \\ &+ 2(I_{i+1}^{SR}I_i^{SR} - I_{i+1}^{HR}I_i^{HR})| + \varepsilon^2\end{aligned}}, \tag{16}$$

where the first line also tries to minimize the distance between SR frames and HR frames in pixel space, while the second line requires the joint product of neighboring SR and HR frames to be the same. In short, through FVL, not only SR frames are required to be close to corresponding HR frames, but their variations are also constrained to be close to that of HR frames. Thus, it guides the generator to recover more temporally consistent videos. Experiments in Section 4.7 validate our proposed frame variation loss.

4 Ablation Experiments

4.1 Training Datasets

For SISR, there are some public datasets of high quality like BSDS 500 [63] and DIV2K [64]. For video SR, a lot of datasets [8], [11], [18] are not available, and MM522 video dataset from [15] has been adopted by recent methods [15], [16], [36] for training. We have made various experiments for ablation studies on MM522 dataset, which contains 522 32-frame video sequences for training collected from 10 videos. Further, we have collected another 20 video sequences for evaluation during training. Following [18], [36], we first apply Gaussian blur to the HR frames, where kernel size is set to 13×13 and the standard deviation is set as 1.6, and then downscale them by sampling every 4th pixel to generate LR frames.

Recently, a lot of methods [17], [41], [65] have also been trained on another video dataset Vimeo-90K [31], which contains 64,612 sequences for training and 7,824 sequences for testing, each of which is composed of 7 frames with the fixed resolution 448×256. In order to verify the generalization ability and robustness of our model, we have also trained and tested our model on Vimeo-90K dataset. As for Vimeo-90K dataset, following [17], [41], [65], we down-sample HR frames bicubically to generate LR frames.

4.2 Implementation Details

We first train our generator network by Charbonnier loss from Equation (14), where ε is empirically set to 10^{-3}. The input LR frames are from various video sequences to form a mini-batch (batch size is 16). Using Adam optimizer [66], we set the initial learning rate at 10^{-3}, and follow polynomial decay to 10^{-4} after 120K iterations, then further down to 10^{-5} gradually. The whole training process lasts for about 250K iterations. We adopt a Leaky ReLU activation [67] after each convolutional layer, with parameter $\alpha = 0.2$.

Then, we apply adversarial training to fine-tune our generator and train a discriminator [40], and loss function for the discriminator is shown in Equation (11). Loss function for our generator is composed of adversarial loss in Equation (12), perceptual loss in Equation (13), and frame variation loss in Equation (15)

$$\mathcal{L}_G = \lambda\mathcal{L}_G^{Ra} + \mathcal{L}_P + \mathcal{L}_{FV}, \tag{17}$$

where λ is set to 5×10^{-3}, and ε in Equation (15) is set to 10^{-3}. We adopt the single-sequence training method, where input LR frames are from one video sequence to generate 16 consecutive SR frames once an iteration. Also using Adam optimizer, we set the initial learning rate at 10^{-4}, and down to 5×10^{-5}, 2.5×10^{-5} and 1×10^{-5}, each for 50K iterations. Unless specified, the input size is 32×32. We conduct experiments on an Intel I7-8700K CPU and one NVIDIA GTX 1080Ti GPU, where ablation studies are performed on TensorFlow platform, and after confirming the best baseline model, we train our network on both TensorFlow and PyTorch platforms. Finally, we introduce GAN into our network on PyTorch platform.

4.3 Non-Local Operation Versus Motion Estimation and Compensation

NLRB is the preprocessing step of our proposed SR network. We first verify its effectiveness by comparing NLRB with ME&MC methods like STMC from [9] and SPMC from [11]. For simplicity, we build a network with 11 convolutional layers (without including merge and magnification module in the tail), where the first convolutional layer uses 5×5 kernel for a large receptive field, while the rest use 3×3 kernel. Besides, residual learning is also adopted to make the training process more stable. Based on this network called VSR, we adopt STMC and SPMC as the preprocessing step respectively to train three models denoted as VSR, VSR-STMC and VSR-SPMC. It is worth noting that, models involving ME&MC also introduce an extra sub-network for motion estimation, thus requiring extra limitation. These models require a motion estimation module for calculating the optical flow

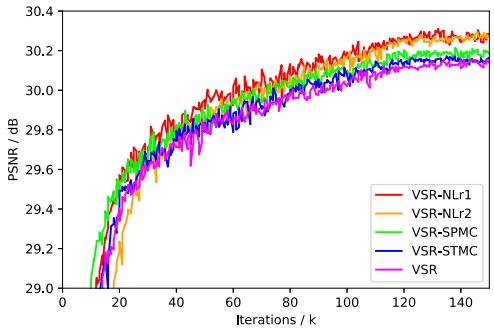


Fig. 7. Training process for different models.

$$F_{i\to j} = (u_{i\to j}, v_{i\to j}) = ME(I_i, I_j; \theta_{ME}), \tag{18}$$

where $F_{i\to j} = (u_{i\to j}, v_{i\to j})$ denotes the optical flow field generated from input frame I_i to I_j, $ME(\cdot)$ represents the operator for calculating optical flow, and θ_{ME} stands for the corresponding parameter. The calculated optical flow $F_{i\to j}$ is later adopted for motion compensation, and the motion estimation is shown as follows:

$$\mathcal{L}_{ME} = \sum_{i=-T}^{T} \left\| I_i^L - \tilde{I}_{0\to i}^L \right\|_1 + \alpha \| \bigtriangledown F_{i\to 0} \|_1, \tag{19}$$

where I_i^L is i_{th} LR frame, $\tilde{I}_{0\to i}^L$ represents the backward warped I_0^L according to optical flow $F_{i\to 0}$, $\bigtriangledown F_{i\to 0}$ denotes the total variation on (u, v) of $F_{i\to 0}$ as described in Equation (18), and α is empirically set as 0.01. The total loss function can be described as

$$\mathcal{L} = \mathcal{L}_C + \lambda \mathcal{L}_{ME}, \tag{20}$$

where $\mathcal{L}_C$ is described in Equation (14), and $\mathcal{L}_{ME}$ denotes the loss of motion estimation sub-network, while λ is empirically set to 0.01.

As illustrated in Fig. 7, VSR-STMC performs very close to the base model VSR, but VSR-SPMC outperforms VSR a little (about 0.04 dB). We further train models with one NLRB, where VSR-NLr1 and VSR-NLr2 set $r = 1$ and $r = 2$ respectively as shown in Fig. 5. It is observed that VSR-NLr1 surpasses the base model VSR (about 0.14 dB), which is superior over VSR-STMC and VSR-SPMC. Note that these two ME&MC methods are rather simple, compared with other sophisticated [41], [68], [69] ones. However, these methods [41], [68], [69] require too many parameters and costly training resources. For instance, FlowNet2 [69] needs more than 100 M parameters so that Wang *et al*. [41] used 8 GPUs for training. Still, sophisticated ME&MC methods may contribute more to the SR task, while they also introduce more calculation complexity. Both STMC and SPMC require extra 53 K parameters for motion estimation, where one NLRB only costs about 14 K parameters. Compared with ME&MC methods, NLRB requires few extra parameters, thus easily inserted into the existing models. So, we do not need an extra loss function to constrain its behavior. Unfortunately, VSR-NLr1 is unable to generate HD video frames on one GPU with only 11 GB memory due to the reasons discussed in Section 3.2, and thus we have to set $r = 2$ for large-size input frames. In contrast, VSR-NLr2 behaves quite close to VSR-NLr1 but is able to handle large-size inputs.

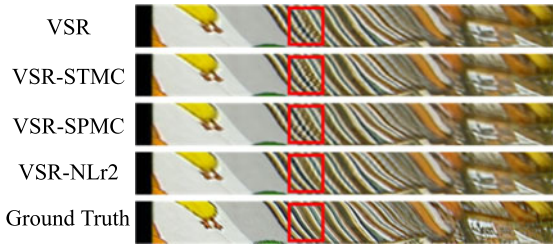


Fig. 8. Temporal profiles of *calendar* from Vid4 [9] dataset.

We make a visual comparison between these models in the form of temporal consistency. A temporal profile is generated by taking the same horizontal row of pixels from consecutive frames and stacking them vertically into a new image [12]. Following VESPCN [9] and FRVSR [12], we extract the temporal profiles and show them in Fig. 8. It can be seen that model VSR, VSR-STMC and VSR-SPMC all generate results with flickering artifacts, while our VSR-NLr2 is able to reconstruct temporally consistent HR frames. Note that although NLRB does not aim at compensating frames to the reference frame, it is able to generate consistent results in good visual quality, which proves its strong ability to capture long-range dependencies.

4.4 Different Fusion Networks

After confirming the validity of NLRB, we next compare the performance and efficiency of networks on various fusion strategies. We first design a base network with our proposed PFRBs without involving multi-scale structure and hybrid convolutions, denoted as PFS. The main body of PFS contains 20 PFRBs, in which every convolutional layer has $N = 32$ filters. The parameter number of PFS is about 4.139 M, and for fairness, we design other networks with a similar number of parameters to compare with PFS. As shown in Fig. 2, there are three main fusion strategies: direct fusion, slow fusion and 3D convolution. Thus, we train three networks adopting these strategies, having them denoted as DFS, SFS and 3DFS respectively. Model DFS also contains 20 residual blocks, each one of which is similar to PFRB but is single-channel. The basic number of convolutional layer filters is set as $N = 85$, in order that PFS and DFS have similar number of parameters. Model SFS is based on DFS, but with an extra merging process before. We still set 20 residual blocks for model 3DFS, each of which is composed of two 3D convolutional layers with $3 \times 3 \times 3$ kernel, where the basic number of convolutional layer filters is set as $N = 60$, due to the same reason above. Note that model PFS, DFS, SFS and 3DFS all employ an NLRB before and a merge and magnification module after their main bodies. As tabulated in Table 1, model PFS, DFS, SFS and 3DFS share a similar number of parameters.

It can be observed from Fig. 9a that, model SFS performs worse than DFS, which accords with [9]. Our model PFS outperforms all other three models, but only a little against 3DFS. However, in spite of similar number of parameters, it is shown in Table 1 that the computational cost of 3DFS is about 28.261 Gflops, which is more than 6 times that of ours (about 4.502 Gflops). We take seven 32×32 LR frames as input to evaluate the computational cost. In addition, we have tested their speeds in handling 480×270 LR frames

TABLE 1
Performance, Parameters, Calculation, and Testing Time Costs of Models With Different Fusion Strategies

Model	DFS	DFS-PS	SFS	SFS-PS	3DFS	3DFS-PS	PFS (ours)	PFS-PS (ours)
Parameter (M)	4.148	0.907	4.174	0.933	4.110	0.998	4.139	0.813
Calculation (Gflops)	4.535	4.535	4.718	4.718	28.261	28.261	4.502	4.502
Testing time (ms)	278	278	293	293	1331	1331	409	409
PSNR (dB)	30.98	30.75	30.86	30.73	31.18	30.82	31.21	31.17

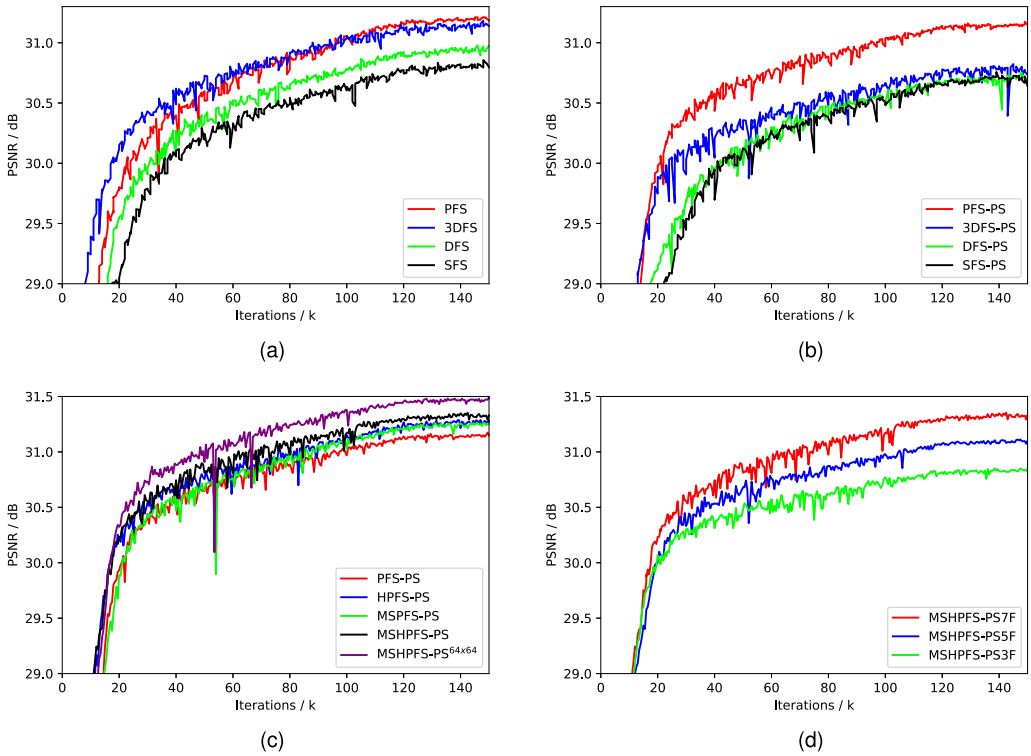


Fig. 9. Various numerical results for different models. (a) Training process for models without parameter sharing. (b) Training process for models with parameter sharing. (c) Training process for models incorporating multi-scale structure and hybrid convolutions. (d) Training process for models with different number of input frames.

under $4\times$ SR. As shown in Table 1, DFS and SFS take about 278 ms and 293 ms to generate one 1920×1080 frame, while 3DFS consumes about 1,331 ms. Our model PFS takes about 409 ms to reconstruct one frame, showing the best performance and relatively fast speed.

As has been discussed before, the multi-channel design of our proposed PFRB makes it possible to adopt a cross-channel parameter sharing scheme. Based on PFS, we train a model with the CCPS scheme, denoted as PFS-PS. For single-channel models like DFS and SFS, we adopt a cross-block parameter sharing (CBPS) scheme from [57] and [58]. As 3DFS already shares the parameters in the temporal dimension, we have to use the CBPS scheme for 3DFS. In practice, as there are 20 residual blocks in DFS, SFS and 3DFS, we consider every 4 residual blocks as one recursive block, and share parameters across these 5 recursive blocks. Although CCPS and CBPS can reduce the number of parameters, they do not decrease the calculation cost and testing time cost. Fig. 9b shows the training process for different models with parameter sharing. It is obvious that DFS-PS, SFS-PS and 3DFS-PS all decline a great deal in the performance, compared with their original counterparts without parameter sharing. However, our model PFS-PS performs only a little worse than PFS (about 0.04 dB) but requires only about 22 percent of the parameters as PFS does.

4.5 Multi-Scale Structure and Hybrid Convolutions

After verifying the effectiveness and efficiency of the proposed progressive fusion strategy, we then confirm the influence of multi-scale structure and hybrid convolutions. We incorporate the multi-scale structure and hybrid convolutions into the PFRB respectively, which are denoted as MSPFS-PS and HPFS-PS. As illustrated in Fig. 9c, MSPFS-PS exceeds the base model PFS-PS by about 0.09 dB, while HPFS-PS achieves an increase of 0.12 dB. Moreover, by combining both the multi-scale structure and hybrid convolutions with our network, the new model MSHPFS-PS surpasses PFS-PS about 0.18 dB. In general, both the multi-scale structure and hybrid convolutions are implemented to enlarge the receptive field of the network, so as to enhance its ability to capture longer-range spatio-temporal correlations.

As mentioned in Section 4.2, we train our models with the input size as 32×32, and whereas, the network may fail to converge to the optimal point as it only has access to a

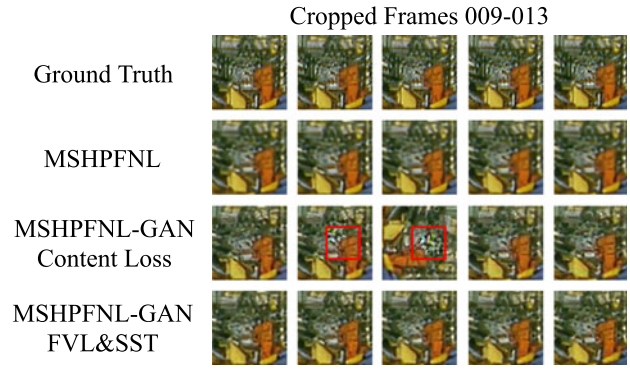


Fig. 10. Cropped frames 009-013 of *calendar* from Vid4 [9] dataset.

small view of input frames. Some recent video SR methods have boosted their performance by training with large-size input (64×64) [17], [41], or with a big mini-batch (64) [37], [38]. Thus, we decide to train MSHPFS-PS with a large input size as 64×64, which is denoted as MSHPFS-PS$^{64\times64}$, and it behaves a lot better than MSHPFS-PS trained with an input size as 32×32. This training strategy makes the network converge to a better point while without introducing extra parameters. However, processing larger size input requires more GPU memory, calculation and time costs. Because we are unable to train a larger model with large-size input on one single GPU, we do not intend to use large-size input for training in Section 5.

4.6 Influence of Input Frame Number

As shown in Fig. 4, our proposed PFRB remains to be a multi-channel structure, whose number of channels is the same as the number of input frames. In Sections 4.3, 4.4, and 4.5, we have explored different models taking 7 frames as input, and we further explore the influence of frame number. Since MSHPFS-PS in Section 4.5 takes 7 frames as input, we denote it as MSHPFS-PS7F. We have trained another two networks adopting 5 and 3 frames as input, which are denoted as MSHPFS-PS5F and MSHPFS-PS3F, respectively. As illustrated in Fig. 9d, the more frames as input, the better performance our network can achieve. This phenomenon accords with common sense, because more frames contain more supplementary information related to the center frame, available online. Besides, since the channel number of a PFRB is the same as the input frame number, more frames also require more calculations, which also contributes to the performance. Note that the calculation cost of a PFRB roughly grows linearly with the input frame number.

4.7 Frame Variation Loss With Single-Sequence Training

After introducing generative adversarial training into video SR, traditional content loss fails to satisfy the demand of reconstructing both realistic and consistent videos. As shown in Fig. 10, we compare 3 models: MSHPFNL without GAN, MSHPFNL-GAN with traditional content loss and MSHPFNL-GAN with FVL&SST. MSHPFNL can only recover rough contours of objects in the frames, albeit MSHPFNL-GAN with traditional content loss is able to reconstruct high-frequency details to a certain extent, it also brings about flickering artifacts and distracting abrupt changes among SR frames. Trained by the proposed FVL and SST method, MSHPFNL-GAN manages to not only recover more realistic details, but also constrain the SR frames to be consistent in temporal. We further provide the temporal profiles of these models in Fig. 11. Evidently, the temporal profile of MSHPFNL is overly smooth, while MSHPFNL-GAN with traditional content loss suffers from a lot of noises and artifacts. In contrast, MSHPFNL-GAN with FVL&SST generates both realistic and consistent results.

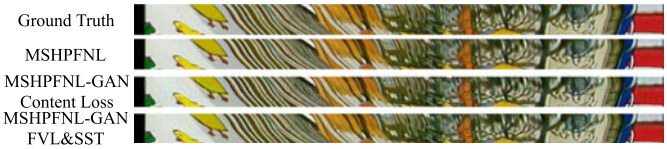


Fig. 11. Temporal profiles of *calendar* from Vid4 [9] dataset.

5 Comparison With State-of-the-Art Methods

Most previous methods [9], [10], [11] down-sample HR frames bicubically to generate LR frames, while recent methods [12], [18] adopt Gaussian blur and then down-sampling scheme. Different down-sampling schemes can determine the LR-to-HR mapping relationship that the network tries to learn. Hence, it is unfair to compare two SR methods under different down-sampling schemes. In order to make a fair comparison with other state-of-the-art methods, we have retrained a lot of methods, e.g., VESPCN [9], DRVSR [11], MMCNN [15], MTUDM [16], DUF_52L [18], RBPN [17] and EDVR [41] by the released codes, using the same training datasets and same down-sampling scheme. Further, we carefully rebuild some non-public methods [10], [12], [53], using the same training datasets and respective training strategies described in their papers. Note that RBPN [17] and EDVR [41] both adopt 8 GPUs for training as described in their papers or provided codes. However, training with so many GPUs is prohibitive for us, so we can only train these two methods with 2 GPUs. Since these two methods use large-size input (64×64) for training that requires large GPU memory, we have to use 32×32 input to train them. As the performance of these methods may be different from that reported in their original papers, we show both performances trained by us and reported in their original papers in Table 2 and Table 4. We train a network with 40 multi-scale hybrid PFRBs and denote it as MSHPFNL to compare with other methods. During training, a data augmentation scheme (random flip and rotation) [49], [70] is used. During testing, neither self-ensemble strategy [70] nor two-stage restoration [41] is adopted for fairness.

5.1 Results on MM522 Training Dataset

As discussed in Section 4.1, We first train our network and other methods on MM522 dataset [15], following down-sampling scheme in [18]. We first conduct experiments on Vid4 video testing dataset [9], which consists of 4 sequences: *calendar*, *city*, *foliage* and *walk*. The PSNR and SSIM [71] are calculated only on the luminance channel of YCbCr colorspace, skipping the first and last two frames and eliminating 8 pixels on four borders of each frame [8]. As shown in Table 2, our model MSHPFNL achieves the best performance, which is

TABLE 2
PSNR (dB) / SSIM of Different Video SR Models on Vid4 Testing Dataset [9] by the Upscaling Factor of 4

Methods	calendar	city	foliage	walk	average	average*
VESPCN [9]	22.20 / 0.7156	26.47 / 0.7246	25.07 / 0.6910	28.40 / 0.8717	25.54 / 0.7507	25.35 / 0.7557
RVSR-LTD [10]	22.07 / 0.7041	26.44 / 0.7217	25.15 / 0.7004	28.29 / 0.8677	25.49 / 0.7485	- / -
MCResNet [53]	22.44 / 0.7319	26.75 / 0.7454	25.30 / 0.7093	28.76 / 0.8788	25.81 / 0.7664	25.45 / 0.7467
DRVSR [11]	22.88 / 0.7586	27.06 / 0.7698	25.58 / 0.7307	29.11 / 0.8876	26.16 / 0.7867	25.52 / 0.7600
FRVSR [12]	23.46 / 0.7854	27.70 / 0.8099	25.96 / 0.7560	29.69 / 0.8990	26.70 / 0.8126	26.69 / 0.8220
MMCNN [15]	23.63 / 0.7969	27.47 / 0.8083	26.01 / 0.7532	29.94 / 0.9030	26.76 / 0.8154	26.28 / 0.7844
MTUDM [16]	23.76 / 0.8026	27.67 / 0.8145	26.08 / 0.7587	30.16 / 0.9069	26.92 / 0.8207	26.57 / 0.7989
DUF_52L [18]	23.85 / 0.8052	27.97 / 0.8253	26.22 / 0.7646	30.47 / 0.9118	27.13 / 0.8267	27.34 / 0.8327
RBPN [17]	24.33 / 0.8244	28.28 / 0.8413	26.46 / 0.7753	30.58 / 0.9130	27.41 / 0.8385	27.16 / 0.8190
EDVR [41]	24.30 / 0.8242	28.04 / 0.8382	26.45 / 0.7744	30.63 / 0.9140	27.36 / 0.8377	27.35 / 0.8264
PFNL [36]	24.37 / 0.8246	28.09 / 0.8385	26.51 / 0.7768	30.64 / 0.9134	27.41 / 0.8383	27.40 / 0.8384
MSHPFNL (ours) -TensorFlow	24.67 / 0.8343	28.55 / 0.8538	26.64 / 0.7825	30.95 / 0.9182	27.70 / 0.8472	27.70 / 0.8472
MSHPFNL (ours) -PyTorch	24.66 / 0.8340	28.55 / 0.8527	26.64 / 0.7828	30.91 / 0.9178	27.69 / 0.8468	27.69 / 0.8468

*Red and blue respectively indicate the best and second-best results. * in the last column denotes the results reported in the original papers.*

TABLE 3
PSNR (dB) / SSIM of Different Video SR Models on UDM10 Testing Dataset [16] by the Upscaling Factor of 4

Methods	archpeople	archwall	auditorium	band	caffe	camera	clap	lake	photography	polyflow	average
VESPCN [9]	35.37/0.9504	40.14/0.9581	27.91/0.8837	33.55/0.9514	37.57/0.9647	43.34/0.9886	34.92/0.9544	30.63/0.8255	35.92/0.9581	36.61/0.9489	35.60/0.9384
RVSR-LTD [10]	35.20/0.9485	39.80/0.9559	27.49/0.8736	33.27/0.9481	37.22/0.9635	43.36/0.9884	34.57/0.9511	30.69/0.8267	35.61/0.9552	36.43/0.9469	35.36/0.9358
MCResNet [53]	35.46/0.9512	40.77/0.9637	27.87/0.8874	33.88/0.9540	38.07/0.9676	43.45/0.9887	35.41/0.9578	30.82/0.8323	36.15/0.9594	37.01/0.9521	35.89/0.9414
DRVSR [11]	35.83/0.9547	41.16/0.9671	29.00/0.9039	34.32/0.9579	39.08/0.9715	45.19/0.9905	36.20/0.9635	31.15/0.8440	36.60/0.9627	37.91/0.9565	36.64/0.9472
FRVSR [12]	36.24/0.9579	41.65/0.9710	29.81/0.9181	34.54/0.9589	39.82/0.9746	46.07/0.9912	36.51/0.9659	31.70/0.8623	36.95/0.9655	38.38/0.9597	37.17/0.9525
MMCNN [15]	36.95/0.9636	42.12/0.9729	30.05/0.9217	35.23/0.9645	40.29/0.9760	46.89/0.9922	37.32/0.9704	31.76/0.8642	37.81/0.9704	38.85/0.9649	37.73/0.9561
MTUDM [16]	37.16/0.9655	42.33/0.9744	30.37/0.9274	35.46/0.9661	40.68/0.9773	47.15/0.9924	37.69/0.9727	32.03/0.8734	38.18/0.9727	39.10/0.9670	38.02/0.9589
DUF_52L [18]	36.92/0.9638	42.53/0.9754	30.27/0.9257	35.49/0.9660	41.03/0.9785	47.30/0.9927	37.70/0.9719	32.06/0.8730	38.02/0.9719	39.25/0.9667	38.05/0.9586
RBPN [17]	38.50/0.9729	43.53/0.9790	31.23/0.9376	35.49/0.9678	41.83/0.9810	49.25/0.9940	38.35/0.9757	32.48/0.8837	38.96/0.9771	40.38/0.9732	39.00/0.9642
EDVR [41]	38.46/0.9732	43.35/0.9783	31.15/0.9372	35.97/0.9696	41.76/0.9808	49.49/0.9947	38.22/0.9759	32.21/0.8790	39.40/0.9793	40.47/0.9739	39.05/0.9642
PFNL [36]	38.35/0.9724	43.55/0.9792	31.18/0.9369	36.01/0.9691	41.84/0.9808	49.26/0.9941	38.33/0.9756	32.53/0.8865	38.95/0.9768	40.04/0.9734	39.00/0.9645
MSHPFNL (ours) -TensorFlow	38.89/0.9750	43.91/0.9807	31.88/0.9444	36.44/0.9717	42.47/0.9824	50.13/0.9948	38.92/0.9780	32.75/0.8923	39.73/0.9802	40.77/0.9763	39.59/0.9676
MSHPFNL (ours) -PyTorch	38.83/0.9748	43.90/0.9806	31.76/0.9434	36.43/0.9715	42.48/0.9824	50.08/0.9948	38.85/0.9777	32.75/0.8919	39.67/0.9799	40.75/0.9763	39.55/0.9673

Red and blue indicate the best and second-best results, respectively.

0.29 dB higher than RBPN [17] in PSNR. Because Vid4 dataset contains only 4 scenes under low resolution (smaller than 720×576), we further conduct experiments on 10 video sequences (1272×720) from [12], [16] for testing. As tabulated in Table 3, our model MSHPFNL still achieves the best performance, which is 0.54 dB higher than EDVR [41] in PSNR. We also train our network on PyTorch platform, which achieves results comparable to that trained on TensorFlow platform.

5.2 Results on Vimeo-90K Training Dataset

Recently, another public video dataset Vimeo-90K has been used by various methods [17], [31], [41], [65]. Likewise, we also train our network on this dataset to prove its generalization ability. Following [17], the PSNR and SSIM [71] are calculated only on the center frame (the 4th frame in a 7-frame sequence), where only the luminance channel of YCbCr colorspace is taken into the calculation, having 8 pixels on four borders eliminated. According to the average motion flow magnitude, Vimeo-90K testing dataset is divided into 3 categories: slow, medium and fast [17], where there are 1,616, 4,983 and 1,225 sequences in each category. Consequently, we give results on these 3 categories separately and on the whole testing dataset. As shown in Table 4, our model MSHPFNL achieves the best performance among the methods trained by us, but it is not as good as the original versions of RBPN [17] and EDVR [41]. This is due to the fact that we train RBPN and EDVR with 2 GPUs, while their original codes require 8 GPUs. As discussed in Section 4.5 and shown in Fig. 9c, more GPUs enable training with larger input size, which is beneficial to the network's convergence.

To further verify the generalization ability of our model, we train dozens of methods on Vimeo-90K dataset, by the Bicubic down-sampling scheme. As shown in Table 4, we also make full comparisons on parameter numbers, testing time and training time of these methods. Our model MSHPFNL requires about 7.770 M parameters while MMCNN and RBPN require more than 10 M parameters and EDVR has more than 20 M parameters. It takes about 1624 ms for

TABLE 4
PSNR (dB) / SSIM of Different Video SR Models on Vimeo-90K Testing Dataset [31] by the Upscaling Factor of 4

Methods	Vimeo-Slow	Vimeo-Medium	Vimeo-Fast	Vimeo-All	Parameter (M)	Test (ms)	Train (h)	GPU	Platform
VESPCN [9]	31.72 / 0.8802	34.07 / 0.9093	37.05 / 0.9332	34.05 / 0.9071	0.106	39	0.9	1	TensorFlow
RVSR-LTD [10]	31.68 / 0.8793	34.08 / 0.9099	36.89 / 0.9319	34.03 / 0.9070	0.371	81	1.5	1	TensorFlow
MCResNet [53]	32.07 / 0.8870	34.58 / 0.9174	37.27 / 0.9350	34.48 / 0.9139	0.230	56	1.3	1	TensorFlow
DRVSR [11]	32.69 / 0.8980	35.39 / 0.9287	37.76 / 0.9411	35.20 / 0.9243	1.722	367	32	1	TensorFlow
FRVSR [12]	32.95 / 0.9015	35.66 / 0.9314	38.01 / 0.9432	35.47 / 0.9271	5.058	187	78	1	TensorFlow
MMCNN [15]	33.09 / 0.9047	35.86 / 0.9340	38.25 / 0.9453	35.66 / 0.9297	10.582	1916	27	1	TensorFlow
MTUDM [16]	33.23 / 0.9071	36.16 / 0.9377	38.55 / 0.9487	35.93 / 0.9331	5.919	1526	21	1	TensorFlow
DUF_52L [18]	33.35 / 0.9092	36.23 / 0.9385	38.41 / 0.9471	35.98 / 0.9338	5.824	2754	51	1	TensorFlow
RBPN [17]	33.69 / 0.9135	36.66 / 0.9418	39.30 / 0.9537	36.46 / 0.9378	12.772	2985 × 2	100	2	PyTorch
EDVR [41]	33.81 / 0.9151	36.71 / 0.9418	39.24 / 0.9524	36.51 / 0.9379	20.699	1166	39	2	PyTorch
PFNL [36]	33.53 / 0.9114	36.40 / 0.9394	38.85 / 0.9495	36.19 / 0.9352	3.003	741	17	1	TensorFlow
MSHPFNL (ours) -TensorFlow	33.95 / 0.9170	36.97 / 0.9446	39.53 / 0.9553	36.75 / 0.9406	7.770	1624	50	1	TensorFlow
MSHPFNL (ours) -PyTorch	33.91 / 0.9164	36.93 / 0.9442	39.47 / 0.9548	36.70 / 0.9401	7.770	1378	35	2	PyTorch
RBPN* [17]	34.18 / 0.9200	37.28 / 0.9470	40.03 / 0.9600	37.07 / 0.9435	12.772	-	-	8	PyTorch
EDVR* [41]	- / -	- / -	- / -	37.61 / 0.9484	20.699	-	-	8	PyTorch

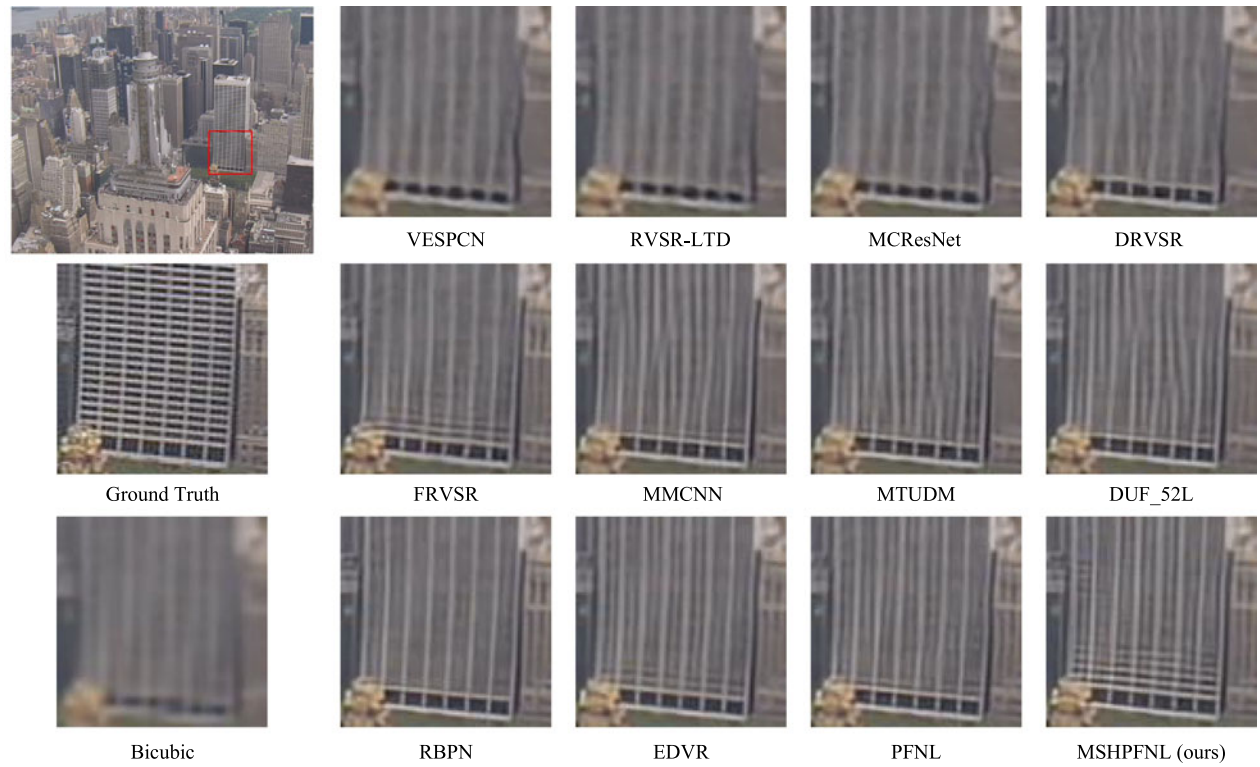
*Red and blue respectively indicate the best and second-best results. * denotes the results reported in the original papers.*

Fig. 12. Visual results of different video SR methods, for $4\times$ upscaling. The frame is from *calendar*.

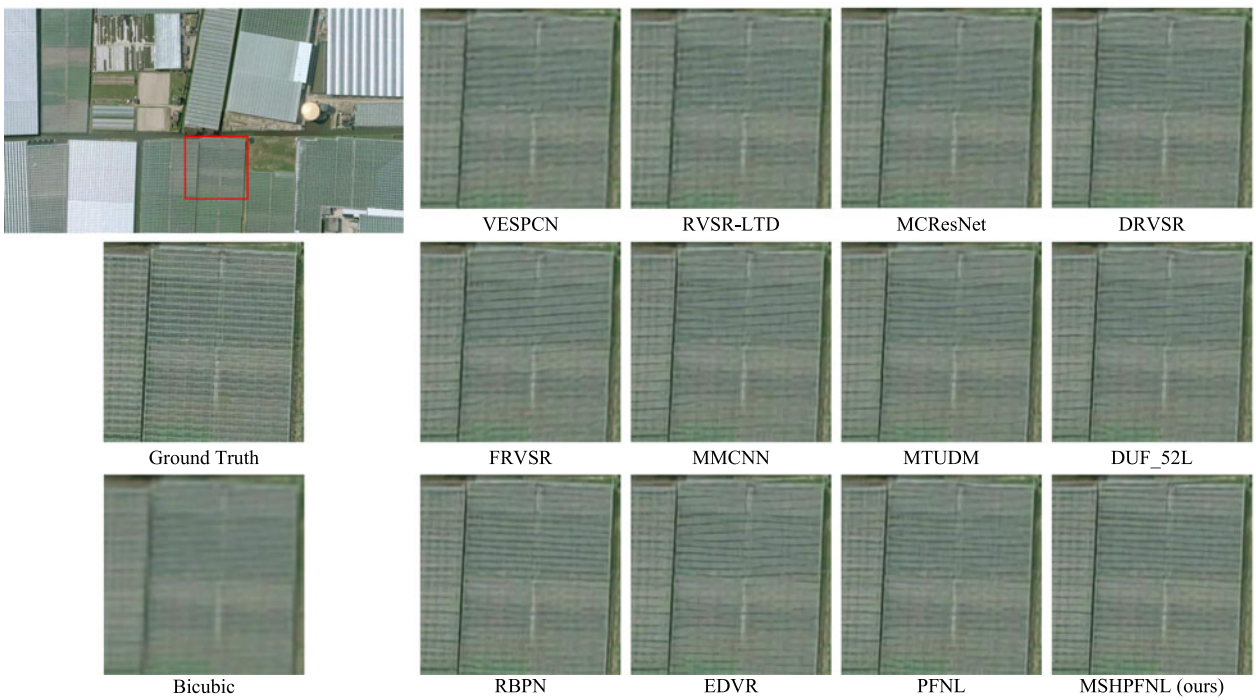
MSHPFNL-TensorFlow and 1378 ms for MSHPFNL-PyTorch to generate one 1920×1080 frame under $4\times$ SR. Note that as RBPN needs too much GPU memory, it is not enough to generate one 1920×1080 frame under $4\times$ SR with one 11 GB GPU. Thus, we calculate the time cost to generate one 960×1080 frame, which is 2985 ms, and we have to speculate that RBPN takes $2985 \times 2 = 5970$ ms for generating one 1920×1080 frame. Moreover, MSHPFNL-TensorFlow spends about 50 hours for training on 1 GPU, and MSHPFNL-PyTorch takes 35 hours on 2 GPUs. FRVSR needs 78 hours on 1 GPU for training, while RBPN takes even 100 hours on 2 GPUs.

5.3 Visual Comparison

As shown in Fig. 12, VESPCN [9], RVSR-LTD [10], MCResNet [53] and DRVSR [11] generate blurry frames without clear characters. Although other methods are able to recover the characters, they fail to recover the smaller vertical stripes in the lower right corner. Only our model MSHPFNL reconstructs both clear characters and vertical stripes. Also shown in Fig. 13a, most other methods result in blurry or twisted superficies of the building, while our model MSHPFNL recovers a clearer surface. We have also conducted experiments on satellite videos, where as shown in Fig. 13b, most



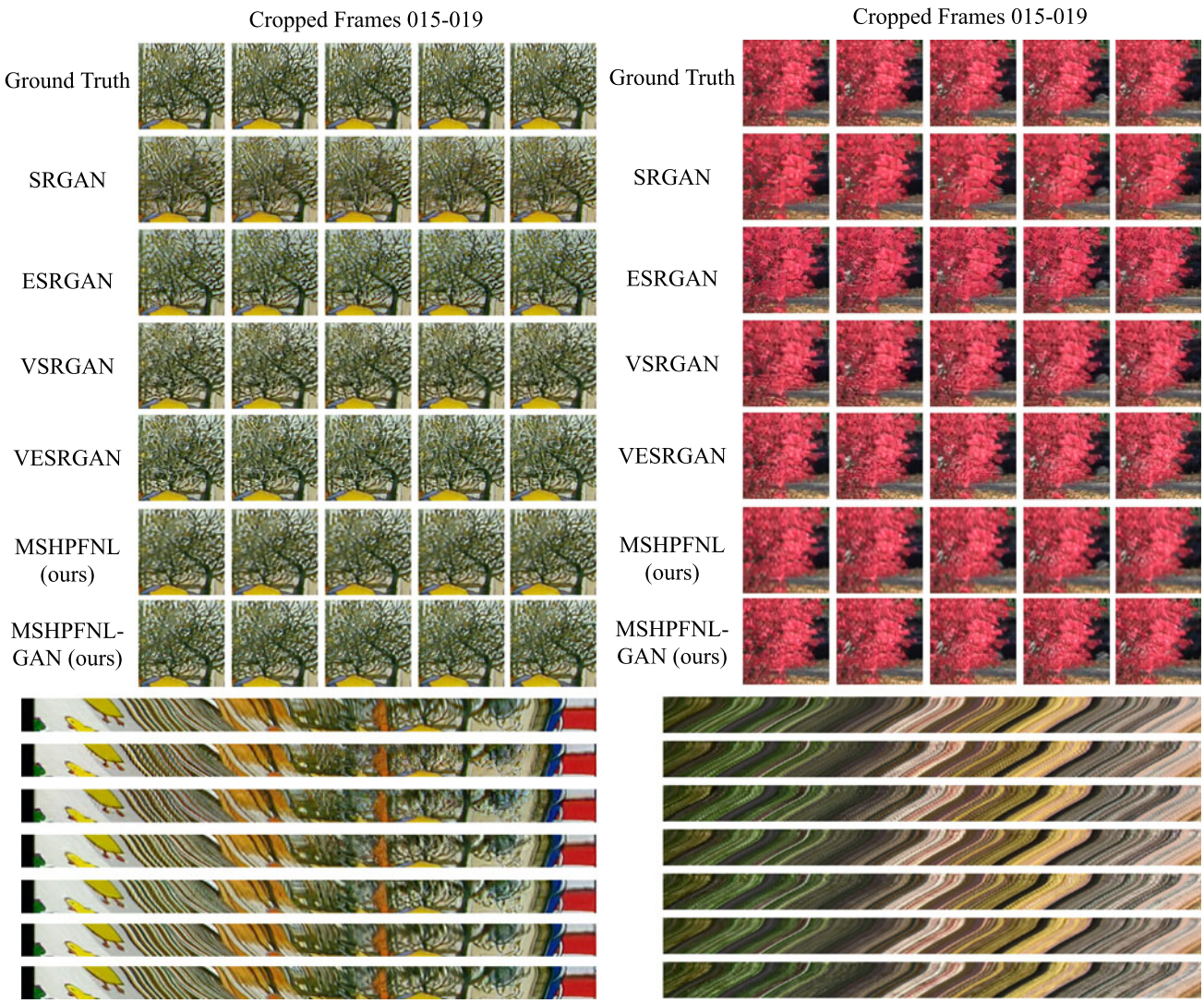
(a) The frame is from *city*



(b) The frame is from a satellite video

Fig. 13. Visual results of different video SR methods, for $4\times$ upscaling.

other methods render blurry and twisted field, while our model MSHPFNL reconstructs field with clearer textures. Lastly, we compare our MSHPFNL-GAN model with SRGAN [39] and ESRGAN [40]. As mentioned in Section 3.3, we apply direct fusion to these 2 models and train their video SR versions, denoted as VSRGAN and VESRGAN. As shown in Fig. 14a, SRGAN and VESRGAN reconstruct many white branches, while ESRGAN generates a lot of blue patterns. Albeit VSRGAN recovers acceptable SR frames, its temporal profile shows that these frames are inconsistent in temporal. MSHPFNL can only generate low-frequency information, while MSHPFNL-GAN reconstructs the most realistic and temporally consistent SR frames. Similar phenomena can be observed in Fig. 14b.



(a) Cropped frames 015-019 and temporal profiles from *calendar* (b) Cropped frames 015-019 and temporal profiles from *foliage*

Fig. 14. Visual results of different GAN based video SR methods, for $4\times$ upscaling.

6 Conclusion

In this paper, we have proposed a novel progressive fusion network that is able to make full use of spatio-temporal information among consecutive frames. We also incorporate the multi-scale structure and a hybrid convolution to increase the receptive field of our network. By adopting a cross-channel parameter sharing strategy, our model performs well even with a small number of parameters. Moreover, we have improved the NLRB suitable for video SR, which captures long-range dependencies directly instead of traditional ME&MC. Based on the above progressive network structure and well-designed modules, we ultimately realized the proposed method under the GAN framework. In particular, we propose a frame variation loss along with the single-sequence training method to fix the unstable phenomenon upon involving GAN into video SR, thereby generating realistic and temporally consistent super-resolved videos. Experimentally, the proposed network substantially outperforms the state-of-the-art methods in both objective metrics and perceptual quality, but with fewer parameters and faster speed.

Acknowledgments

This work was supported by the National Key R&D Project (2016YFE0202300), National Natural Science Foundation of China (61671332, U1903214, U1736206, 62071339, 62072350, 62072347, 61971165, and 61773295), and Hubei Province Technological Innovation Major Project (2019AAA049, and 2019AAA045).

References

[1] L. Zhou, Z. Wang, Y. Luo, and Z. Xiong, "Separability and compactness network for image recognition and superresolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3275–3286, Nov. 2019.

[2] X. Yu, B. Fernando, R. Hartley, and F. Porikli, "Semantic face hallucination: Super-resolving very low-resolution face images with supplementary attributes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 11, pp. 2926–2943, Nov. 2020.

[3] K. Jiang, Z. Wang, P. Yi, G. Wang, T. Lu, and J. Jiang, "Edge-enhanced GAN for remote sensing image superresolution," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5799–5812, Aug. 2019.

[4] Z. Shao, L. Wang, Z. Wang, and J. Deng, "Remote sensing image super-resolution using sparse representation and coupled sparse autoencoder," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 12, no. 8, pp. 2663–2674, Aug. 2019.

[5] X. Liu, D. Zhao, R. Xiong, S. Ma, W. Gao, and H. Sun, "Image interpolation via regularized local linear regression," *IEEE Trans. Image Process.*, vol. 20, no. 12, pp. 3455–3469, Dec. 2011.

[6] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.

[7] C. Dong, C. L. Chen, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[8] A. Kappeler, S. Yoo, Q. Dai, and A. K. Katsaggelos, "Video super-resolution with convolutional neural networks," *IEEE Trans. Comput. Imag.*, vol. 2, no. 2, pp. 109–122, Jun. 2016.

[9] J. Caballero *et al.*, "Real-time video super-resolution with spatio-temporal networks and motion compensation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2848–2857.

[10] D. Liu *et al.*, "Robust video super-resolution with learned temporal dynamics," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2526–2534.

[11] X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia, "Detail-revealing deep video super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4482–4490.

[12] M. S. M. Sajjadi, R. Vemulapalli, and M. Brown, "Frame-recurrent video super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6626–6634.

[13] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7794–7803.

[14] Y. Huang, W. Wang, and L. Wang, "Bidirectional recurrent convolutional networks for multi-frame super-resolution," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 235–243.

[15] Z. Wang *et al.*, "Multi-memory convolutional neural network for video super-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 5, pp. 2530–2544, May 2019.

[16] P. Yi, Z. Wang, K. Jiang, Z. Shao, and J. Ma, "Multi-temporal ultra dense memory network for video super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 8, pp. 2503–2516, Aug. 2020.

[17] M. Haris, G. Shakhnarovich, and N. Ukita, "Recurrent back-projection network for video super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3897–3906.

[18] Y. Jo, S. W. Oh, J. Kang, and S. J. Kim, "Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3224–3232.

[19] A. Lucas, S. López-Tapia, R. Molina, and A. K. Katsaggelos, "Generative adversarial networks and perceptual losses for video super-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3312–3327, Jul. 2019.

[20] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3D residual networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 5534–5542.

[21] J. Shuiwang, Y. Ming, X. Wei, and Y. Kai, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[22] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. Int. Conf. Learn. Representations*, 2016.

[23] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[24] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense convolutional networks for efficient prediction," in *Proc. Int. Conf. Learn. Representations*, 2018.

[25] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[26] K. Jiang *et al.*, "Multi-scale progressive fusion network for single image deraining," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8346–8355.

[27] S. P. Belekos, N. P. Galatsanos, and A. K. Katsaggelos, "Maximum a posteriori video super-resolution using a new multichannel image prior," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1451–1464, Jun. 2010.

[28] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar, "Fast and robust multiframe super resolution," *IEEE Trans. Image Process.*, vol. 13, no. 10, pp. 1327–1344, Oct. 2004.

[29] R. Liao, X. Tao, R. Li, Z. Ma, and J. Jia, "Video super-resolution via deep draft-ensemble learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 531–539.

[30] C. Liu and D. Sun, "On Bayesian adaptive video super resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 346–60, Feb. 2014.

[31] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *Int. J. Comput. Vis.*, vol. 127, no. 8, pp. 1106–1125, 2019.

[32] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2005, vol. 2, pp. 60–65.

[33] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.

[34] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 11, pp. 2599–2613, Nov. 2019.

[35] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 294–310.

[36] P. Yi, Z. Wang, K. Jiang, J. Jiang, and J. Ma, "Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3106–3115.

[37] T. Isobe *et al.*, "Video super-resolution with temporal group attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8008–8017.

[38] Y. Tian, Y. Zhang, Y. Fu, and C. Xu, "TDAN: Temporally-deformable alignment network for video super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3360–3369.

[39] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 105–114.

[40] X. Wang *et al.*, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 63–79.

[41] X. Wang, K. C. Chan, K. Yu, C. Dong, and C. C. Loy, "EDVR: Video restoration with enhanced deformable convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 1954–1963.

[42] C. Dong, C. L. Chen, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 391–407.

[43] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1874–1883.

[44] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4809–4817.

[45] K. Zhang, W. Zuo, and L. Zhang, "Learning a single convolutional super-resolution network for multiple degradations," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3262–3271.

[46] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *Proc. Int. Conf. Learn. Representations*, 2019.

[47] J. Ma, X. Wang, and J. Jiang, "Image superresolution via dense discriminative network," *IEEE Trans. Ind. Electron.*, vol. 67, no. 7, pp. 5687–5695, Jul. 2020.

[48] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep Laplacian pyramid networks for fast and accurate super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5835–5843.

[49] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2472–2481.

[50] Z. Zhang, X. Wang, and C. Jung, "DCSR: Dilated convolutions for single image super-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 4, pp. 1625–1635, Apr. 2019.

[51] W. Zhang, Y. Liu, C. Dong, and Y. Qiao, "RankSRGAN: Generative adversarial networks with ranker for image super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3096–3105.

[52] T. R. Shaham, T. Dekel, and T. Michaeli, "SinGAN: Learning a generative model from a single natural image," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4570–4580.

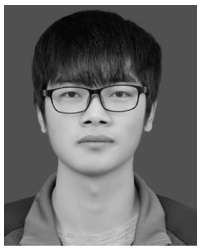
[53] D. Li and Z. Wang, "Video superresolution via motion compensation and deep residual learning," *IEEE Trans. Comput. Imag.*, vol. 3, no. 4, pp. 749–762, Dec. 2017.

[54] X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 802–810.

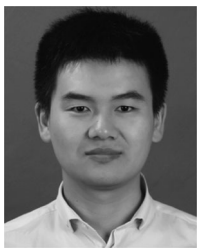
[55] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1664–1673.
[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[57] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1637–1645.
[58] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2790–2798.
[59] A. Vaswani *et al.*, "Attention is all you need," *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[60] J. Johnson, A. Alahi, and L. Feifei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.
[61] A. Jolicoeurmartineau, "The relativistic discriminator: A key element missing from standard GAN," in *Proc. Int. Conf. Learn. Representations*, 2019.
[62] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: http://arxiv.org/abs/1409.1556
[63] P. A. Arbelaez, M. Maire, C. C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.
[64] R. Timofte *et al.*, "NTIRE 2017 challenge on single image super-resolution: Methods and results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1110–1121.
[65] W. Bao, W. Lai, X. Zhang, Z. Gao, and M. Yang, "MEMC-Net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Sep. 17, 2019, doi: 10.1109/TPAMI.2019.2941941.
[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2014.
[67] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2016, pp. 1026–1034.
[68] A. Dosovitskiy *et al.*, "FlowNet: Learning optical flow with convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2758–2766.
[69] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1647–1655.
[70] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1132–1140.
[71] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

Peng Yi received the BS degree from the Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, China, in 2017. He is currently working toward the PhD degree under the supervision of Prof. Zhongyuan Wang in the School of Computer Science, Wuhan University, China.

Zhongyuan Wang (Member, IEEE) received the PhD degree in communication and information system from Wuhan University, Wuhan, China, in 2008. He is currently a professor at the School of Computer Science, Wuhan University, Wuhan, China. He has been directing four projects funded by the National Natural Science Foundation of China. He has authored or coauthored more than 80 refereed journal and conference papers and has been granted more than 30 invention patents. His research interests include biometrics and computer vision.

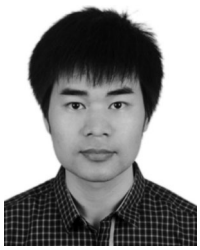


Kui Jiang received the BS degree from Xinjiang University, China, in 2017, and the MS degree from Wuhan University, China, in 2019. He is currently working toward the PhD degree under the supervision of Prof. Zhongyuan Wang in the School of Computer Science, Wuhan University, China.



Junjun Jiang (Member, IEEE) received the BS degree from the Department of Mathematics, Huaqiao University, Quanzhou, China, in 2009, and the PhD degree from the School of Computer, Wuhan University, Wuhan, China, in 2014. From 2015 to 2018, he was an associate professor with the China University of Geosciences, Wuhan, China. Since 2016, he has been a project researcher with the National Institute of Informatics, Tokyo, Japan. He is currently a professor at the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China. His research interests include image processing and computer vision. He was a recipient of the Finalist of the World's FIRST 10K Best Paper Award at ICME 2017, the Best Student Paper Runner-up Award at MMM 2015, the 2016 China Computer Federation (CCF) Outstanding Doctoral Dissertation Award, and the 2015 ACM Wuhan Doctoral Dissertation Award.

Tao Lu received the BS and MS degrees from the School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan, China, in 2003 and 2008, respectively, and the PhD degree from the National Engineering Research Center for Multimedia Software, Wuhan University, China, in 2013. He is currently a professor at the School of Computer Science and Engineering, Wuhan Institute of Technology, China, and also a research member with the Hubei Provincial Key Laboratory of Intelligent Robot. He held a postdoctoral position with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, Texas, from 2015 to 2017. His research interests include image/video processing, computer vision, and artificial intelligence.



Jiayi Ma (Member, IEEE) received the BS degree in information and computing science and the PhD degree in control science and engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2008 and 2014, respectively. He is currently a professor at the Electronic Information School, Wuhan University, China. He has authored or coauthored more than 140 refereed journal and conference papers, including the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, the *IEEE Transactions on Image Processing*, the *International Journal of Computer Vision*, CVPR, ICCV, ECCV, *etc.* His research interests include computer vision, machine learning, and pattern recognition. He has been identified in the 2019 Highly Cited Researchers list from the Web of Science Group. He is also an area editor of the *Information Fusion*, an associate editor of the *Neurocomputing*, and a guest editor of the *Remote Sensing*.