Progressive Fusion Video Super-Resolution Network via Exploiting Non-Local Spatio-Temporal Correlations

Peng Yi[1], Zhongyuan Wang*[1], Kui Jiang[1], Junjun Jiang[2,3], and Jiayi Ma[1]

[1]Wuhan University, Wuhan, China [2]Harbin Institute of Technology, Harbin, China
[3]Peng Cheng Laboratory, Shenzhen, China

{yipeng, kuijiang}@whu.edu.cn, {wzy_hope, junjun0595}@163.com, jyma2010@gmail.com

Abstract

Most previous fusion strategies either fail to fully utilize temporal information or cost too much time, and how to effectively fuse temporal information from consecutive frames plays an important role in video super-resolution (SR). In this study, we propose a novel progressive fusion network for video SR, which is designed to make better use of spatio-temporal information and is proved to be more efficient and effective than the existing direct fusion, slow fusion or 3D convolution strategies. Under this progressive fusion framework, we further introduce an improved non-local operation to avoid the complex motion estimation and motion compensation (ME&MC) procedures as in previous video SR approaches. Extensive experiments on public datasets demonstrate that our method surpasses state-of-the-art with 0.96 dB in average, and runs about 3 times faster, while requires only about half of the parameters.

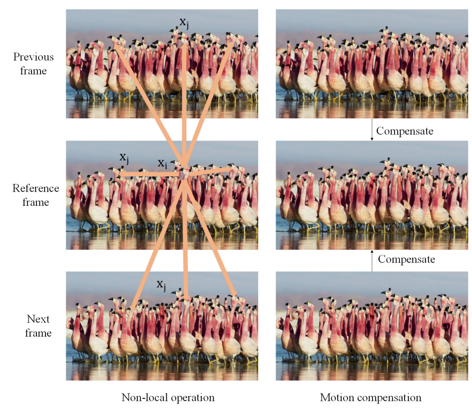


Figure 1: Differences of non-local operation and ME&MC. Non-local operation tries to obtain the response at position x_i by computing the weighted average of relationships of all possible positions x_j [36]. ME&MC tries to compensate neighboring frames to the reference frame.

1. Introduction

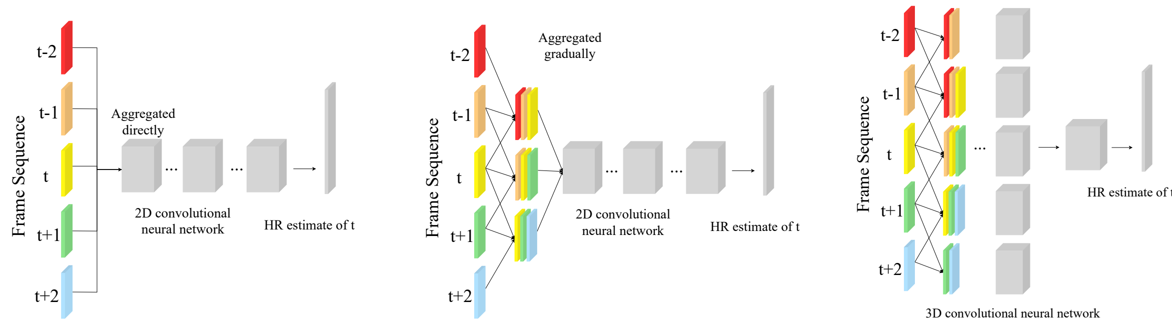
Super-resolution refers to the problem of reconstructing a high-resolution (HR) output from a given low-resolution (LR) input. For single image super-resolution (SISR), as the input consists of only one single image, it is concentrated on extracting spatial information. For video SR, where the input is composed of several consecutive frames, it emphasizes exploiting the inter-frame temporal correlations. Thus, it remains to be a non-trivial problem on making full use of temporal correlations among multiple frames.

In general, most video SR methods adopt a time-based way or space-based way for utilizing temporal information. Time-based methods [11, 26, 31] consider frames as time-series data, and send frames through the network one by one. However, this kind of approach may run slower as it cannot process multiple frames in parallel.

Space-based methods [3, 13, 14] take multiple frames as *supplementary materials to help reconstruct the reference frame*. Compared to time-based methods, this kind of approach is able to retain full inter-frame temporal correlations and enjoy the advantages of parallel computing. Most existing space-based approaches adopt direct fusion, slow fusion or 3D convolution [25, 29] for temporal information fusion [3], which are shown in Figure 2.

Most traditional video SR methods [2, 7, 20, 22] conduct pixel-level motion and blur kernel estimation based on image priors, trying to solve a complex optimization problem [23]. With recent development of deep learning, a lot of video SR methods [3, 14, 23, 26, 31] based on convolutional neural networks (CNNs) have emerged. They try to conduct

*Corresponding author. Code will be available at https://github.com/psychopa4/PFNL.



(a) Direct fusion. (b) Slow fusion. (c) 3D convolution.

Figure 2: Different merging schemes 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.

explicit motion compensation on the input frames to make it easier for the SR network to capture long-range dependencies. Although these alignment methods are intended for increasing the temporal coherence, they have two main disadvantages: 1) they introduce extra parameters, calculations and training difficulty, and 2) incorrect ME&MC may corrupt original frames and decrease the performance of SR. We conduct experiments to find that these ME&MC [3, 31] methods contribute little (about 0.01-0.04 dB) to the video SR in Section 4.2.

Recently, Wang *et al*. proposed non-local neural networks [36] for capturing long-range dependencies, which essentially shares similar purpose as the ME&MC in video SR. As illustrated in Figure 1, 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.

In this study, we propose a novel progressive fusion network by incorporating a series of *progressive fusion residual blocks* (PFRBs). The proposed PFRB is intended for *better utilization of spatio-temporal information from multiple frames*. Besides, multi-channel design of the PFRB makes it possible to behave well even under few parameters by adopting a kind of parameter sharing strategy. Moreover, we introduce and improve the non-local residual block (NLRB) to capture long-range spatio-temporal correlations directly. We elaborate these two modules in Section 3.

2. Related Work

Since the pioneer's works by Dong *et al*. [4, 5], a lot of CNN-based SR methods [8, 24, 27, 33, 40] have been proposed. Following time-based way, DRVSR [31] adopts convolutional long short-term memory (ConvLSTM) [28] module in the network, reserving information from the last frame. Based on spatial transformer motion compensation (STMC) [3], DRVSR also proposes a novel sub-pixel motion compensation (SPMC) layer, projecting LR frames onto HR image space. Another time-based method FRVSR [26] first sends one LR frame through the network to obtain a super-resolved HR output. This HR output is concatenated to the next LR frame, flowing through the network again to obtain the corresponding HR estimate. By this way, as frames go through the network, their outputs are sent back to help reconstruct following frames. However, FRVSR can only utilize one previous frame to help reconstruct current LR frame, neglecting the potential of next LR frames.

Space-based methods [3, 13, 14, 23] try to merge temporal information in a parallel manner, and three main fusion strategies are shown in Figure 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. VSRnet [14] adopts direct fusion, and DUFVSR [13] builds a 3D CNN, while VESPCN [3] has explored these three strategies for video SR. Our proposed progressive fusion strategy also follows the space-based way, thus, we have conducted experiments to compare progressive fusion against these three strategies, which are elaborated in Section 4.

3. Our Method

3.1. Progressive Fusion Network

We first introduce the designs of our whole network. As illustrated in Figure 3, we first adopt a non-local block to process LR frames. Then, after one 5×5 convolution layer, we add 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

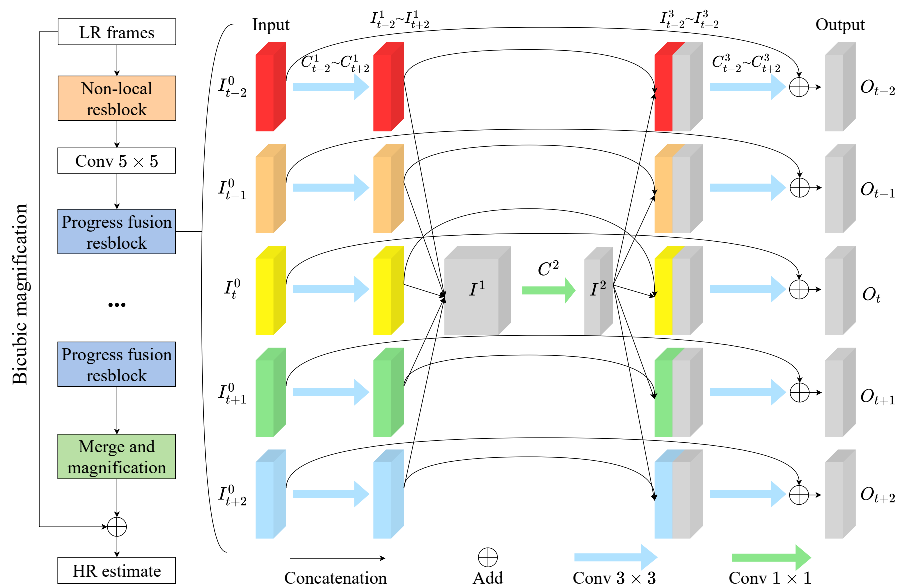


Figure 3: The architecture of our whole network (left) and the structure of one PFRB (right). Note that the input number of the residual block can be arbitrary, and we show the case of taking five frames as input. The "$\bigoplus$" represents element-wise add. Black, blue and green arrows denote concatenation, convolution with 3×3 and 1×1 kernel respectively.

residual HR image, which is added to the bicubically magnified image to obtain the HR estimate as in [15]. Merge and magnification module applies sub-pixel magnification layer from [27]. The whole processing procedure is quite straightforward, nevertheless, we have designed two sophisticated residual blocks for better performance.

We show the structure of a PFRB in the right of Figure 3, when taking five frames as input. Basically, we follow a multi-channel design methodology, because the input contains multiple LR frames. We first conduct 3×3 convolutional layers with same certain number (N) of filters, which can be described as:

$$I_t^1 = C_t^1(I_t^0), \tag{1}$$

where t denotes the index of temporal dimension, I_t^0 represents the input frames, and $C_t^1(\cdot)$ is the first depth of the convolutional layers. I_t^1 denotes the feature maps extracted, which are supposed to contain rather *self-independent features from each input frame*. Later, feature maps $\{I_{t-2}^1, ..., I_{t+2}^1\}$ are concatenated and merged into one part, which contains information from all input frames, and the depth of this merged feature maps is $5 \times N$ when taking five frames as input. That is to say, this aggregated deep feature map contains *a large deal of temporal-correlated information*. Naturally, we use one convolutional layer to further utilize 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 is 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 is *most temporal-correlated to the center frame*. This process can be described as:

$$I^2 = C^2(I^1) = C^2(\{I_{t-2}^1, ..., I_{t+2}^1\}), \tag{2}$$

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, this distillated feature map I^2 is concatenated to all the previous feature maps $\{I_{t-2}^1, ..., I_{t+2}^1\}$, becoming $\{I_{t-2}^3, ..., I_{t+2}^3\}$. By this way, feature maps of all channels contain two kinds of information now: *self-independent spatial information* and *fully mixed temporal information*. 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:

$$O_t = C_t^3(I_t^3) + I_t^0 = C_t^3(\{I^2, I_t^1\}) + I_t^0, \tag{3}$$

where I_t^3 denotes the merged feature maps $\{I^2, I_t^1\}$, $C_t^3(\cdot)$ represents the third depth of the convolutional layers, and O_t is the corresponding output. We use "$+I_t^0$" to represent residual learning [9], which means that the output and the input are required to have the same size. Thus, the third depth of convolutional layers have N filters, for the purpose of 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, in which *both intra-frame spatial correlations and inter-frame temporal correlations are fully extracted progressively*. Although the parameter number of one PFRB can grow linearly with the number of input frames, we further adopt a parameter sharing scheme to reduce the parameters with barely no loss in performance. Different from the schemes in [16] and [30], where they share the parameters across different residual blocks, we conduct parameter sharing across channels within one residual block. We also conduct experiments comparing our network against direct fusion, slow fusion and 3D convolution strategies with different parameter sharing schemes, which are shown in Section 4.3.

3.2. Non-Local Residual Block

Inspired from [34, 36], we introduce non-local neural networks to capture long-range dependencies through a kind of non-local operation. Mathematically, the non-local operation is described as follows:

$$y_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j) g(x_j), \tag{4}$$

where x represents the input data (image, video, text, etc), y denotes the output, which should be with 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 $g(\cdot)$ gives a representation of the input. $C(x)$ is used for normalization.

According to Equation (4), it can be inferred that this non-local operation tries to obtain a representation at one position by computing the correlations between it and all possible positions, which is the exact reason we choose it to replace traditional ME&MC. Wang *et al*. [36] 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 Figure 4, the output of a 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 [9]. We first reshape $\boldsymbol{X}$ to $\boldsymbol{X_1}$, transforming the temporal dimension into channel dimension. That is to say, *the temporal correlations is not calculated directly, but captured through the channel correlations.* If not doing so, the shape of feature map $\boldsymbol{F}$ would be $TWH \times TWH$, which is the same as [36]. 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 [27], we reshape $\boldsymbol{X_1}$ to $\boldsymbol{X_2}$, reducing the input size to deepen its channels, and r denotes the down-scaling factor. By that means, the shape of feature map $\boldsymbol{F}$ becomes $HW/r^2 \times HW/r^2$, which is not related to number of frames T anymore. In this way, it can enable our model to generate large-size video frames. For $4\times$ video SR, we choose $r = 2$ to make our network capable of reconstructing full HD 1920×1080 video frames (480×270 LR frames as input). A NLRB does not change the input shape, thus, it can be inserted into the existed models conveniently. We show the superiority of the NLRB against ME&MC in Section 4.2.

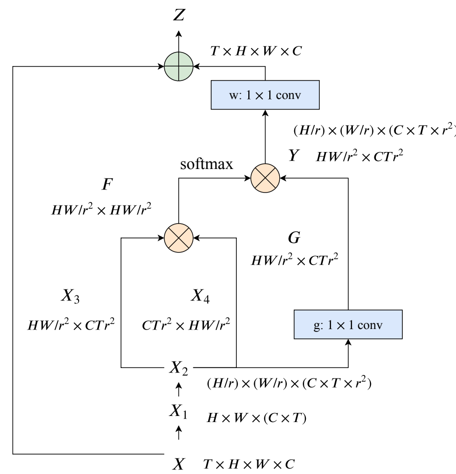


Figure 4: Structure of one NLRB. We show the feature maps as their shapes like $T \times H \times W \times C$, and they are reshaped if noted. The "$\bigotimes$" represents matrix multiplication and "$\bigoplus$" represents element-wise add.

4. Experiments

4.1. Implementation Details

For SISR, there are some datasets with high quality like BSDS 500 [1] and DIV2K [32]. For video SR, a lot of datasets [13, 14, 31] are not available, and we adopt a public video dataset from [38] for training. This video dataset contains 522 video sequences for training, which are collected from 10 videos. Further, we have collected another 20 video sequences for evaluation during training. Following [13], 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 4-th pixel to generate LR frames. During training, we set the batch size as 16, and input LR frame size as 32×32. We choose Charbonnier loss function (a differentiable variant of $\mathcal{L}_1$ norm) [18] to train our network:

$$\mathcal{L}_{SR} = \sqrt{(H - SR(I))^2 + \varepsilon^2}, \tag{5}$$

where I represents input LR frames, H is the corresponding HR center frame, $SR(\cdot)$ denotes the function of the super-resolution network, and ε is empirically set to 10^{-3}. Using Adam optimizer [17], we set initial learning rate at 10^{-3}, and follow polynomial decay to 10^{-4} after 120K iterations, then further down to 10^{-5} gradually. We adopt a Leaky ReLU activation [10] after each convolutional layer, with parameter $\alpha = 0.2$. We conduct experiments with an Intel I7-8700K CPU and one NVIDIA GTX 1080Ti GPU.

4.2. Non-Local Operation vs Motion Estimation and Compensation

NLRB is the preprocessing step of our proposed SR network, and we first verify its effectiveness by comparing NLRB with ME&MC methods like STMC from [3] and SPMC from [31]. For simplicity, we build a network with 11 convolutional layers (not including merge and magnification module in the tail), where the first convolutional layer uses 5×5 kernel for a big 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 preprocessing step respectively, and train three models denoted as VSR, VSR-STMC and VSR-SPMC. It is worth mentioning that, models involving ME&MC also introduce an extra sub-network for motion estimation, thus requiring extra limitation. The total loss function can be described as:

$$\mathcal{L} = \mathcal{L}_{SR} + \lambda \mathcal{L}_{ME}, \tag{6}$$

where $\mathcal{L}_{SR}$ is described in Equation (5), and $\mathcal{L}_{ME}$ denotes the loss of motion estimation sub-network, which is specifically described in [3, 31], while λ is empirically set as 0.01.

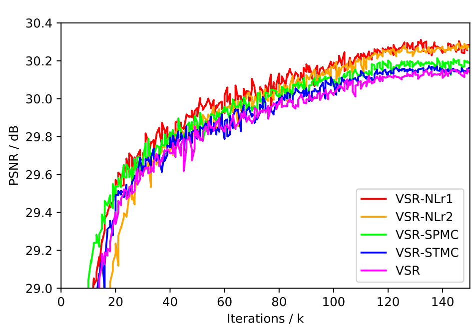


Figure 5: Training processes for different models.

As illustrated in Figure 5, VSR-STMC performs very close to the base model VSR, and VSR-SPMC outperforms VSR little (about 0.04 dB). We further train models adopting one NLRB, where VSR-NLr1 and VSR-NLr2 set $r = 1$ and $r = 2$ respectively as shown in Figure 4. It is observed that VSR-NLr1 surpasses the base model VSR (about 0.14 dB), which is superior to VSR-STMC and VSR-SPMC. It is worth mentioning that these two ME&MC methods are rather simple, compared to other sophisticated [6, 12, 35] ones. However, these methods [6, 12, 35] are too complex, for instance, Wang *et al*. [35] adopt 8 GPUs for training. Still, sophisticated ME&MC methods may contribute more to the SR task, but they also introduce more calculation complexity. Both STMC and SPMC require extra 53 K parameters for motion estimation, while one NLRB only costs about 14 K parameters. Compared to ME&MC methods, NLRB requires little extra parameters and is easy to be inserted into the existed models, and we do not need extra loss function to limit its behavior. Unfortunately, VSR-NLr1 is not able to generate HD video frames on one GPU with only 11 GB memory due to the reasons discussed in Section 3.2, thus, we have to set $r = 2$ for large-size input frames. Fortunately, VSR-NLr2 behaves quite close to VSR-NLr1 and is able to handle large-size inputs.

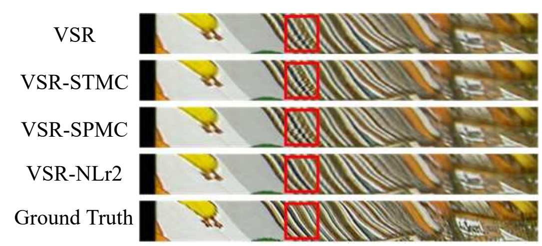
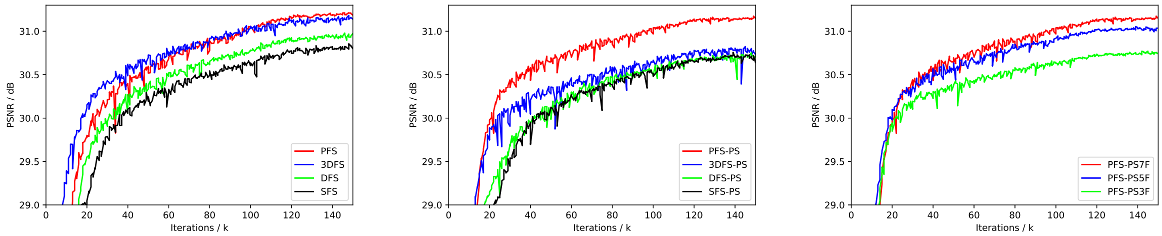


Figure 6: Temporal profiles of *calendar* from Vid4 [3] dataset.

We make 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 [26]. Following VESPCN [3] and FRVSR [26], we extract the temporal profiles and show them in Figure 6. It can be observed that, model VSR, VSR-STMC and VSR-SPMC all generate results with flickering artifacts, while our VSR-NLr2 is able to reconstruct temporal-consistent HR frames. Note that although NLRB is not aiming at compensating frames to the reference frame, it is able to generate consistent results with good visual quality, which also proves its strong ability to capture long-range dependencies.

4.3. Different Fusion Networks

After confirming the validity of NLRB, we compare the performances and efficiencies of networks adopting various fusion strategies. We first design a base network adopting our proposed PFRBs, which is denoted as PFS. The main



(a) (b) (c)

Figure 7: Various numerical results for different models. (a) Training processes for models without parameter sharing. (b) Training processes for models with parameter sharing. (c) Training processes for models with different number of input frames.

Table 1: Performances, parameters, calculation and testing time cost of different models.

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

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 *similar number of parameters* to compare with PFS. As demonstrated in Figure 2, there are three main fusion strategies: direct fusion, slow fusion and 3D convolution. Thus, we train three networks adopting these strategies, and they are denoted as DFS, SFS and 3DFS respectively. Model DFS also contains 20 residual blocks, and each one of which is similar to PFRB but is single-channel. The basic number of convolutional layer filters is set as $N = 85$, for the purpose of making 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, and 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 adopt a NLRB before and a merge and magnification module after their main bodies. As demonstrated in Table 1, model PFS, DFS, SFS and 3DFS share similar number of parameters.

It can be observed from Figure 7(a) that, model SFS behaves worse than DFS, which accords with [3]. Our model PFS surpasses all other three models, but only out performs 3DFS a little. However, although with similar number of parameters, it is shown in Table 1 that the calculation 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 compute the calculation cost. Besides, we have tested their speeds for handling 480×270 LR 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 needs about 1331 ms. Our model PFS takes about 409 ms to reconstruct one frame with 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 (CCPS) scheme. Based on PFS, we train a model adopting the CCPS scheme, which is denoted as PFS-PS. For single-channel models like DFS and SFS, we adopt a cross-block parameter sharing (CBPS) scheme from [16] and [30]. As 3DFS already shares the parameters in the temporal dimension, we have to adopt 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. Figure 7(b) shows the training processes for different models adopting parameter sharing. It is obvious that DFS-PS, SFS-PS and 3DFS-PS all decline a great deal in the performance, compared to their original versions without parameter sharing. However, our model PFS-PS performs only a little worse than PFS (about 0.04 dB) but requires only about 22% the parameters as PFS does.

Table 2: PSNR (dB) / SSIM of different video SR models on Vid4 testing dataset, when upscaling factor is 4. Red and blue indicate the best and second-best performance. * in the last row denotes the performances reported in the original papers.

Sequences	VESPCN [3]	RVSR-LTD [23]	MCResNet [19]	DRVSR [31]	FRVSR [26]	DUF_52L [13]	PFNL (ours)
calendar	22.21 / 0.7160	21.91 / 0.6915	22.42 / 0.7308	22.88 / 0.7586	23.44 / 0.7846	23.85 / 0.8052	24.37 / 0.8246
city	26.48 / 0.7257	26.32 / 0.7124	26.75 / 0.7456	27.06 / 0.7698	27.65 / 0.8047	27.97 / 0.8253	28.09 / 0.8385
foliage	25.07 / 0.6913	25.07 / 0.6922	25.31 / 0.7091	25.58 / 0.7307	25.97 / 0.7529	26.22 / 0.7646	26.51 / 0.7768
walk	28.40 / 0.8719	28.24 / 0.8663	28.74 / 0.8784	29.11 / 0.8876	29.70 / 0.8991	30.47 / 0.9118	30.65 / 0.9135
average	25.54 / 0.7512	25.38 / 0.7406	25.81 / 0.7660	26.16 / 0.7867	26.69 / 0.8103	27.13 / 0.8267	27.40 / 0.8384
average*	25.35 / 0.7557	- / -	25.45 / 0.7467	25.52 / 0.7600	26.69 / 0.8220	27.34 / 0.8327	27.40 / 0.8384

Table 3: PSNR (dB) / SSIM of different video SR models, when upscaling factor is 4. Red and blue indicate the best and second-best performance.

Sequences	VESPCN [3]	RVSR-LTD [23]	MCResNet [19]	DRVSR [31]	FRVSR [26]	DUF_52L [13]	PFNL (ours)
archpeople	35.37 / 0.9504	35.22 / 0.9488	35.45 / 0.9510	35.83 / 0.9547	36.20 / 0.9577	36.92 / 0.9638	38.35 / 0.9724
archwall	40.15 / 0.9582	39.90 / 0.9554	40.78 / 0.9636	41.16 / 0.9671	41.96 / 0.9713	42.53 / 0.9754	43.55 / 0.9792
auditorium	27.90 / 0.8837	27.42 / 0.8717	27.92 / 0.8877	29.00 / 0.9039	29.81 / 0.9168	30.27 / 0.9257	31.18 / 0.9369
band	33.54 / 0.9514	33.20 / 0.9471	33.85 / 0.9538	34.32 / 0.9579	34.53 / 0.9584	35.49 / 0.9660	36.01 / 0.9692
caffe	37.58 / 0.9647	37.02 / 0.9624	38.04 / 0.9675	39.08 / 0.9715	39.77 / 0.9743	41.03 / 0.9785	41.87 / 0.9809
camera	43.36 / 0.9886	43.58 / 0.9888	43.35 / 0.9885	45.19 / 0.9905	46.02 / 0.9912	47.30 / 0.9927	49.26 / 0.9941
clap	34.92 / 0.9544	34.54 / 0.9511	35.40 / 0.9578	36.20 / 0.9635	36.52 / 0.9646	37.70 / 0.9719	38.32 / 0.9756
lake	30.63 / 0.8257	30.62 / 0.8232	30.82 / 0.8323	31.15 / 0.8440	31.53 / 0.8489	32.06 / 0.8730	32.53 / 0.8865
photography	35.94 / 0.9582	35.57 / 0.9548	36.13 / 0.9592	36.60 / 0.9627	37.06 / 0.9656	38.02 / 0.9719	39.00 / 0.9770
polyflow	36.62 / 0.9490	36.38 / 0.9452	36.98 / 0.9520	37.91 / 0.9565	38.29 / 0.9581	39.25 / 0.9667	40.05 / 0.9735
average	35.60 / 0.9384	35.34 / 0.9348	35.87 / 0.9414	36.64 / 0.9472	37.17 / 0.9507	38.05 / 0.9586	39.01 / 0.9645

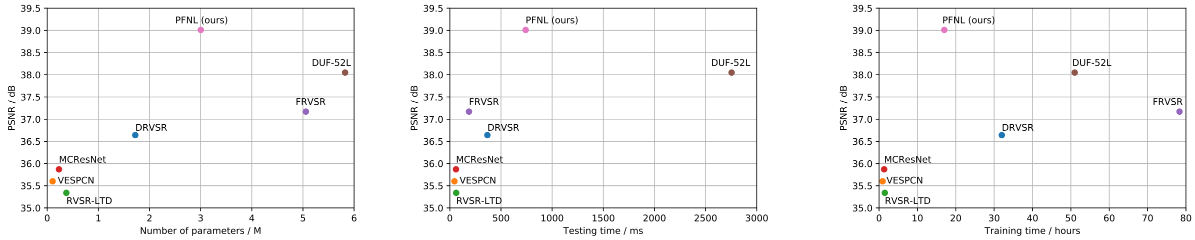
4.4. Influence of Input Frame Number

As has been shown in Figure 3, our proposed PFRB remains to be a multi-channel design, whose number of channels is the same as the number of input frames. In Sections 4.2 and 4.3, we have explored different models taking 7 frames as input, and we here further explore the influence of frame number. Since PFS-PS in Section 4.3 takes 7 frames as input, we denote it as PFS-PS7F there. We have trained another two networks adopting 5 and 3 frames as input, and they are denoted as PFS-PS5F and PFS-PS3F respectively. As illustrated in Figure 7(c), it can be observed that 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. Besides, since the channel number of a PFRB is the same as the input frame number, more frames also require more calculations, which also contribute to increase the performance. Note that the calculation cost of a PFRB basically grows linearly with the input frame number.

4.5. Comparison with state-of-the-art Methods

Most previous methods [3, 23, 31] down-sample HR frames bicubically to generate LR frames, while recent methods [13, 26] adopt Gaussian blur and then down-sampling scheme. Different down-sampling schemes can affect the LR-to-HR mapping relationship the network tries to learn, thus, 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 re-trained a lot of methods, *e.g.* VESPCN [3], DRVSR [31], and DUF_52L [13] by their provided codes, using the same training datasets, with same down-sampling scheme and Tensorflow platform. Further, we carefully rebuild some non-public methods [19, 23, 26], using the same training datasets and different training strategies described in their papers. Still, performances of these methods may be different from that reported in their original papers. Table 2 gives both the performances achieved by us and reported in thier origiant papers. Based on PFS-PS, we set the basic convolutional layer filter number as 64, and denote this network as PFNL to compare with other methods. During training, a data agumentation scheme (random flip and rotation) from [21, 41] is adopted.

We first conduct experiments on Vid4 video testing datasets from [3], which consists of 4 sequences: *calendar*, *city*, *foliage* and *walk*. The PSNR and SSIM [37] are calculated only on luminance channel of YCbCr colorspace, skipping the first and last two frames and eliminating 8 pix-



(a) (b) (c)

Figure 8: (a) Parameter numbers and performances of various methods. (b) Testing time cost (generating one 1920×1080 frame when upscaling factor is 4) and performances. (c) Training time cost and performances.

(a) Ground truth (b) VESPCN (c) RVSR-LTD (d) MCResNet (e) DRVSR (f) FRVSR (g) DUF_52L (h) PFNL (ours)

(a) This frame is from *auditorium*

(a) Ground truth (b) VESPCN (c) RVSR-LTD (d) MCResNet (e) DRVSR (f) FRVSR (g) DUF_52L (h) PFNL (ours)

(b) This frame is from *photography*

Figure 9: Visual results of different video SR methods, for $4\times$ upscaling.

els on four borders of each frame [14]. Our model PFNL achieves the best performance, which is shown in Table 2. Because Vid4 dataset only contains 4 scenes under low resolution (smaller than 720×576), we further collect 10 frame sequences (1272×720) from [26, 39] for testing. As demonstrated in Table 3, our model PFNL outperforms DUF_52L about 0.96 dB in average. Moreover, as shown in Figure 8, we make full comparisons on parameter numbers, testing time and training time cost of these methods. Our model PFNL requires about 3.003 M parameters while DUF_52L needs about 5.829 M parameters. It takes about 741 ms for PFNL to generate one 1920×1080 frame under $4\times$ SR, while DUF_52L spends about 2754 ms. PFNL spends about 17 hours for training, and DUF_52L needs 51 hours, while FRVSR can take 78 hours.

As illustrated in Figure 9(a), most other methods generate blurry or misleading numbers, while our method recovers the right numbers with good visual quality. Also shown in Figure 9(b), most other methods generate blurry texture with wrong direction, while our method is able to reconstruct clear and right texture.

5. Conclusion

In this paper, we propose a novel progressive fusion network that is able to make full use of spatio-temporal information among consecutive frames. We have introduced and improved the NLRB suitable for video SR, capturing long-range dependencies directly instead of adopting traditional ME&MC. The proposed network is able to outperform the state-of-the-art methods with fewer parameters and faster speed.

Acknowledgements

This study is supported by National Key R&D Project (2016YFE0202300), National Natural Science Foundation of China (61671332, U1736206, 61971165), Hubei Province Technological Innovation Major Project (2017AAA123, 2019AAA049) and Natural Science Fund of Hubei Province (2018CFA024).

References

[1] Pablo Andres Arbelaez, Michael Maire, Charless C Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[2] Stefanos P. Belekos, Nikolaos P. Galatsanos, and Aggelos K. Katsaggelos. Maximum a posteriori video super-resolution using a new multichannel image prior. *IEEE Transactions on Image Processing*, 19(6):1451–1464, 2010.

[3] Jose Caballero, Christian Ledig, Andrew Peter Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2848–2857, 2017.

[4] Chao Dong, Change Loy Chen, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[5] Chao Dong, Change Loy Chen, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European Conference on Computer Vision (ECCV)*, pages 391–407, 2016.

[6] Alexey Dosovitskiy, Philipp Fischery, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.

[7] Sina Farsiu, Michael D Robinson, Michael Elad, and Peyman Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, 2004.

[8] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Deep back-projection networks for super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1664–1673, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2016.

[11] Yan Huang, Wei Wang, and Liang Wang. Bidirectional recurrent convolutional networks for multi-frame super-resolution. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 235–243, 2015.

[12] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1647–1655, 2017.

[13] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3224–3232, 2018.

[14] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K. Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE Transactions on Computational Imaging*, 2(2):109–122, 2016.

[15] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016.

[16] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645, 2016.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

[18] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5835–5843, 2017.

[19] Dingyi Li and Zengfu Wang. Video superresolution via motion compensation and deep residual learning. *IEEE Transactions on Computational Imaging*, 3(4):749–762, 2017.

[20] Renjie Liao, Xin Tao, Ruiyu Li, Ziyang Ma, and Jiaya Jia. Video super-resolution via deep draft-ensemble learning. In *IEEE International Conference on Computer Vision (ICCV)*, pages 531–539, 2015.

[21] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1132–1140, 2017.

[22] Ce Liu and Deqing Sun. On bayesian adaptive video super resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(2):346–60, 2014.

[23] Ding Liu, Zhaowen Wang, Yuchen Fan, Xianming Liu, Zhangyang Wang, Shiyu Chang, and Thomas Huang. Robust video super-resolution with learned temporal dynamics. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2526–2534, 2017.

[24] Jiayi Ma, Xinya Wang, and Junjun Jiang. Image super-resolution via dense discriminative network. *IEEE Transactions on Industrial Electronics*, 2019.

[25] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3D residual networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5534–5542, 2017.

[26] Mehdi S. M Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6626–6634, 2018.

[27] Wenzhe Shi, Jose Caballero, Ferenc Huszr, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution

using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[28] Xingjian Shi, Zhourong Chen, Hao Wang, Wang Chun Woo, Wang Chun Woo, and Wang Chun Woo. Convolutional lstm network: a machine learning approach for precipitation nowcasting. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 802–810, 2015.

[29] Ji Shuiwang, Yang Ming, Xu Wei, and Yu Kai. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2012.

[30] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2798, 2017.

[31] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4482–4490, 2017.

[32] Radu Timofte, Kyoung Mu Lee, Xintao Wang, Yapeng Tian, Ke Yu, Yulun Zhang, Shixiang Wu, Chao Dong, Liang Lin, and Yu Qiao. Ntire 2017 challenge on single image super-resolution: Methods and results. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1110–1121, 2017.

[33] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4809–4817, 2017.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Llion Jones, Jakob Uszkoreit, Aidan N Gomez, and ukasz Kaiser. Attention is all you need. *International Conference on Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[35] Xintao Wang, Kelvin C.K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[36] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2018.

[37] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[38] Zhongyuan Wang, Peng Yi, Kui Jiang, Junjun Jiang, Zhen Han, Tao Lu, and Jiayi Ma. Multi-memory convolutional neural network for video super-resolution. *IEEE Transactions on Image Processing*, 28(5):2530–2544, 2019.

[39] Peng Yi, Zhongyuan Wang, Kui Jiang, Zhenfeng Shao, and Jiayi Ma. Multi-temporal ultra dense memory network for video super-resolution. *IEEE Transactions on Circuits and Systems for Video Technology*, 2019.

[40] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3262–3271, 2018.

[41] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, 2018.