Video Super-Resolution via Bidirectional Recurrent Convolutional Networks

Yan Huang, Wei Wang, and Liang Wang, *Senior Member, IEEE*

Abstract—Super resolving a low-resolution video, namely video super-resolution (SR), is usually handled by either single-image SR or multi-frame SR. Single-Image SR deals with each video frame independently, and ignores intrinsic temporal dependency of video frames which actually plays a very important role in video SR. Multi-Frame SR generally extracts motion information, e.g., optical flow, to model the temporal dependency, but often shows high computational cost. Considering that recurrent neural networks (RNNs) can model long-term temporal dependency of video sequences well, we propose a fully convolutional RNN named bidirectional recurrent convolutional network for efficient multi-frame SR. Different from vanilla RNNs, 1) the commonly-used full feedforward and recurrent connections are replaced with weight-sharing convolutional connections. So they can greatly reduce the large number of network parameters and well model the temporal dependency in a finer level, i.e., patch-based rather than frame-based, and 2) connections from input layers at previous timesteps to the current hidden layer are added by 3D feedforward convolutions, which aim to capture discriminate spatio-temporal patterns for short-term fast-varying motions in local adjacent frames. Due to the cheap convolutional operations, our model has a low computational complexity and runs orders of magnitude faster than other multi-frame SR methods. With the powerful temporal dependency modeling, our model can super resolve videos with complex motions and achieve well performance.

Index Terms—Deep learning, recurrent neural networks, 3D convolution, video super-resolution

1 Introduction

Since a large number of high-definition devices have sprung up, generating high-resolution videos from original low-resolution content, namely video super-resolution (SR), is under great demand. Recently, various methods have been proposed to handle this problem, which can be classified into two categories: 1) single-image SR [7], [14], [15], [24], [27], [46], [52] super resolves each low-resolution video frame independently, and 2) multi-frame SR [2], [3], [32], [32], [34], [38] super resolves a low-resolution video frame by jointly considering its multiple adjacent frames as the input and especially modeling the temporal dependency among them.

The temporal dependency is usually considered as an essential component of video SR, which critically affects the final quality of SR. Such dependency often occurs in two different aspects: long-term and short-term along the time axis. The long-term dependency means that all video frames at different timesteps share the similar visual content, e.g., the background of video frames usually remains constant or varies smoothly with the movement of camera view. While the short-term dependency mainly refers to sudden motions of partial visual objects during only a few adjacent frames, e.g., the moving human leg in a video where a man is running. Compared with the long-term dependency that focuses on the modeling of constant or slow-varying content, the short-term dependency emphasizes on the understanding of fast-varying motions.

Existing multi-frame SR methods generally model the long-term and short-term dependencies by indistinguishably extracting subpixel motions of video frames, e.g., estimating optical flow based on sparse prior integration or variation regularity [2], [32], [34]. But such motion estimation can only be effective for video sequences that contain slow-varying motions. Most multi-frame SR methods [3], [13], [38] also make a strict assumption that the underlying motion has a simple analytic form, e.g., the blur kernel is known, which oversimplifies the complex nature of real-world video sequences. To overcome these problems, several solutions have been explored by avoiding the explicit motion estimation [37], [44]. Unfortunately, they still have to perform implicit motion estimation to reduce the temporal aliasing, and use additional resolution enhancement when fast-varying motions are encountered. In addition, the computational cost of these multi-frame SR methods is usually very high. The computational bottleneck lies in the accurate motion estimation, e.g., taking about two hours when super resolving a 720×480 frame [32]. Such high computational cost dramatically limits the real-world application of these methods.

- *Y. Huang and W. Wang are with the Center for Research on Intelligent Perception and Computing (CRIPAC), National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA), and the University of Chinese Academy of Sciences (UCAS), Huairou, Beijing 100049, China. E-mail: {yhuang, wangwei}@nlpr.ia.ac.cn.*
- *L. Wang is with the Center for Research on Intelligent Perception and Computing (CRIPAC), National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA), and the University of Chinese Academy of Sciences (UCAS), Huairou, Beijing 100049, China, and the Center for Excellence in Brain Science and Intelligence Technology (CEBSIT), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100864, China. E-mail: wangliang@nlpr.ia.ac.cn.*

Manuscript received 11 Oct. 2016; revised 9 Mar. 2017; accepted 1 Apr. 2017. Date of publication 3 May 2017; date of current version 13 Mar. 2018.
(Corresponding Author: Yan Huang)
Recommended for acceptance by A.Vedaldi.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2017.2701380

A class of deep models named recurrent neural networks (RNNs) [49] have shown their great power of temporal dependency modeling in sequential data processing, e.g., action recognition [8] and image-sentence matching [23]. In this paper, we explore the use of RNNs for dealing with the temporal dependency in video SR. A straightforward approach in this direction is to treat given low-resolution frames as sequential inputs to a RNN, and then regard its outputs at different timesteps as predicted high-resolution frames. However, such an approach is infeasible resulting from three major problems. 1) Since both feedforward and recurrent connections of the RNN are fully connected, and the number of pixels in a video frame is usually very large, e.g., tens of thousands, the RNN will contain millions of learning parameters. These parameters will dramatically increase the model complexity and make the model very hard to train. 2) The hidden states of RNN are 1D vectors that are globally transformed from input 2D low-resolution frames. Such dimensionality reduction inevitably loses the structure information of original frames which plays a crucial role in the prediction of high-resolution frames. 3) Recurrent connections operating on hidden states can only be capable of modeling global slow-varying motions but not those short-term fast-varying ones. Because the important fine-grained details for characterizing fast-varying motions mostly exist in raw frames rather than hidden states [2].

To deal with these issues, we propose a bidirectional recurrent convolutional network (BRCN) for efficient multi-frame SR. The proposed network can be regarded as a fully convolutional version of RNNs, which replaces the commonly-used full feedforward and recurrent connections with weight-sharing convolutional connections. In this way, learning parameters of BRCN only include filter weights and biases. Then the number of parameters can be largely reduced from millions to a few tens of thousands. The hidden states now are 2D feature maps convolved by filters rather than 1D vectors as vanilla RNNs. These feature maps can naturally preserve the structure information of edge direction and relative position in the visual content of original video frames.

To extract the important fine-grained details from raw frames for modeling fast-varying motions, the feedforward connections in BRCN are 3D convolutions rather than conventional 2D ones. Particularly, this scheme is implemented by feeding multiple input layers at the previous timesteps to the current hidden layer in a convolutional manner. So the hidden layer is able to capture informative patterns along both spatial and temporal dimensions, to describe the motions occurring at the same location across a series of frames. Different from recurrent convolutions that mainly deal with long-term slow-varying motions, 3D feedforward convolutions focus on enhancing the understanding of fast-varying motions. They directly operate on original frames that can provide more visual details than abstracted hidden layers. In fact, these two types of convolutions can cooperatively model the temporal dependency in a comprehensive way due to their complementary characteristics.

The direction of vanilla RNNs is usually forward along the timeline, which models the dependency relation between the current frame and its previous frames. To additionally consider the dependency relation between the current frame and its future frames, our BRCN combines a forward and a backward sub-networks to jointly make the final prediction. We apply the proposed model to super resolve videos with complex motions. The experimental results demonstrate its efficiency and effectiveness by achieving magnitude faster speed than other multi-frame SR methods, as well as better performance compared with state-of-the-art methods.

Our main contributions can be summarized as follows.

- We propose a bidirectional recurrent convolutional network for multi-frame SR, where the temporal dependency can be efficiently modelled by recurrent and 3D feedforward convolutions.
- It is an end-to-end framework which does not need pre-/post-processing. Our convolutions can scale to videos with any spatial size and temporal step.
- We achieve better performance and faster speed than existing multi-frame SR methods.

The rest of the paper is organized as follows. In Section 2, we briefly review related work. In Section 3, we detail the proposed model. In Section 4, we apply the proposed model to the task of video SR. Finally, we conclude the paper in Section 5.

2 Related Work

We review related work from the following perspectives.

Single-Image SR. Irani and Peleg [24] propose the primary work for this problem, followed by Freeman et al. [14] studying it in a learning-based way. To alleviate high computational complexity, Bevilacqua et al. [5] and Chang et al. [7] introduce manifold learning techniques which can reduce the required number of image patch exemplars. For further acceleration, Timofte et al. [46] propose the anchored neighborhood regression method. Yang et al. [52] and Zeyde et al. [53] exploit compressive sensing to encode image patches with a compact dictionary and obtain sparse representations. To make the dictionary cover large appearance variations, Huang et al. [20] expand the internal patch search space by allowing geometric variations. Considering that sparse coding based methods ignore the consistency of pixels in overlapped patches, Gu et al. [16] develop a convolutional sparse coding based SR method.

In addition, there exist some state-of-the-art methods based on convolutional neural networks (CNNs). Dong et al. [9], [10], [11] learn a conventional CNN for single-image SR which achieves good performance as well as fast speed. Kim et al. [28] develop a residual CNN and find that increasing the depth of networks can greatly improve the final accuracy. Shi et al. [42] design an efficient sub-pixel CNN that can perform real-time SR on a single K2 GPU. Kim et al. [29] propose a deeply-recursive CNN for single-image SR which obtains state-of-the-art results. In this work, we focus on multi-frame SR by modeling temporal dependency which is more suitable for dealing with video sequences.

Multi-Frame SR. Tsai and Huang [48] propose the seminal work of multi-frame SR, which is then extensively studied in [36]. Some early works [3], [17], [38] perform simple motion estimation with affine models. To suit the nature of complex motions in real world videos, Baker and Kanade [2] extract optical flow as motion estimation for video SR. Then, various improvements [6], [32], [33], [34] around this work are explored to better handle complex motions. However, these methods all suffer from the high computational cost due to

the accurate motion estimation. To deal with this problem, Protter et al. [37] and Takeda et al. [44] avoid motion estimation by employing nonlocal mean and 3D steering kernel regression. Unfortunately, these methods still need to estimate pixel-wise motions when encountering large motions. Liao et al. [30] generate multiple SR drafts from input low-resolution videos and use a deep CNN for ensemble learning. It can exploit contextual information of motion estimation provided from external data for video SR. While we propose bidirectional recurrent and 3D feedforward convolutions as an alternative to efficiently model the temporal dependency that can achieve much faster speed and well performance

Deep Learning. Since the year of 2006, many deep learning models have been proposed and applied to various applications [18], [22]. Our work is related to some work that is built on CNN and RNN as follows. Jain and Seung [25] and Eigen et al. [12] demonstrate the effectiveness of CNN for image denoising with certain patterns. Ji et al. [26] extend convolutions from 2D input images to 3D videos, which learn both spatial and temporal features simultaneously and show effectiveness on a variety of video tasks [47]. Schusterand and Paliwal [39] propose a bidirectional RNN to model temporal dependency in both forward and backward directions. Shi et al. [50] design a convolutional version of long short-term memory RNN (LSTM-RNN) [19] for accurate precipitation nowcasting. Different from them, our proposed model can be regarded as a fully convolutional version of RNN, which includes especially designed recurrent and 3D feedforward convolutions for efficient temporal dependency modeling in the task of video SR.

It should be noted that this paper is a systematic extension of our preliminary conference version [21]. In previous wok, we only empirically find that the modeling of fixed temporal contexts (by conditional convolution) can improve the performance. While in this submission, we reframe the previous 2D feedforward and conditional convolution jointly as a 3D feedforward convolution, allowing the model to flexibly get access to varying temporal contexts in a natural way. Under this more general framework, the previous work can just be regarded as a special case when the temporal step of 3D feedforward convolution is 2. More importantly, we present considerable new explanations on: 1) how the 3D feedforward convolution can suitably handle short-term fast-varying motions by modeling temporal contexts, and 2) how it can cooperate with the recurrent convolution for temporal dependency modeling in a complementary way.

In addition, we perform more experiments to verify the effectiveness of our model as follows. 1) We demonstrate the optimal model architecture by testing a variety of BRCNs with different filter sizes of recurrent convolution, temporal lengths of training volume, directions of network and temporal steps of 3D feedforward convolution. 2) We comprehensively evaluate the performance of our model by performing experiments on more datasets, measuring our results with more criteria, super resolving videos using more upscaling factors, making comparisons with more recent methods, and presenting more visualization results. 3) We extend our model to simultaneously process three color channels rather than luminance channel only, and show that our model can outperform or be comparable to state-of-the-art methods but run orders of magnitude faster than them.

3 Bidirectional Recurrent Convolutional Network

3.1 Formulation

Given a low-resolution, noisy and blurry video, our goal is to obtain a high-resolution, noise-free and blur-free version. We propose a bidirectional recurrent convolutional network (BRCN) to map the low-resolution frames to high-resolution ones. As shown in Fig. 1, the proposed network contains a forward sub-network and a backward sub-network to model the temporal dependency from both previous and future frames. The two sub-networks of BRCN are denoted by two black blocks with dash borders, respectively. In the forward sub-network, there are four layers including the input layer, the first hidden layer, the second hidden layer and the output layer. The states of all these layer are all 2D feature maps rather 1D vectors as vanilla RNNs. These feature maps are connected by two types of convolutions:

3D Feedforward Convolution. The 3D feedforward convolutions denoted by black arrows (in Fig. 1) connect not only the input layer at the current timestep but also multiple adjacent layers at the previous timesteps to the current hidden layer. The details of a 3D feedforward convolution are shown in Fig. 2, where its temporal step is 3 denoted by yellow, red and black lines. By regarding the previous input layers as the contextual information for the current input layer, the convolution can extract discriminate representations in both spatial and temporal dimensions to describe short-term fast-varying motions occurring in these input layers. The temporal step of 3D feedforward convolution should not be very large, since fast movements usually occur in local adjacent frames.

Recurrent Convolution. The recurrent convolutions denoted by blue arrows (in Fig. 1) connect hidden layers of two adjacent frames, where the inference of the current hidden layer is conditioned on the hidden layer at the previous timestep. The details of recurrent convolution are shown in Fig. 2, denoted by blue lines. The filter weights of recurrent convolution are shared among all the timesteps, so such an autoregressive scheme is able to capture repetitive patterns of transformation between pairwise hidden layers within a long range. But different from 3D feedforward convolutions, 2D recurrent convolutions operate on a more abstract level as hidden layers rather than detailed frames, and therefore are more suitable for capturing global slow-varying patterns.

We use these two convolutions in both forward and backward sub-networks, with the goal can make full use of the forward and backward temporal dynamics. We denote the frame set of a low-resolution video[1] $\mathcal{X}$ as $\{\mathbf{X}_i\}_{i=1,2,\ldots,T}$, and infer the other three layers as follows.

First Hidden Layer. When inferring the first hidden layer $\mathbf{H}^f_{1,i}$ (or $\mathbf{H}^b_{1,i}$) at the ith timestep in the forward (or backward) sub-network, two different inputs are considered: 1) the stacked ith frame and its previous (or future) t_{w_1} ones along the time axis, denoted by $[\mathbf{X}]^f_i$ (or $[\mathbf{X}]^b_i$), connected by a 3D feedforward convolution, and 2) the hidden layer $\mathbf{H}^f_{1,i-1}$ (or $\mathbf{H}^b_{1,i+1}$) at the $i-1$th (or $i+1$th) timestep connected by a recurrent convolution:

1. Note that we upscale each low-resolution frame in a video sequence to the desired size with bicubic interpolation in advance.

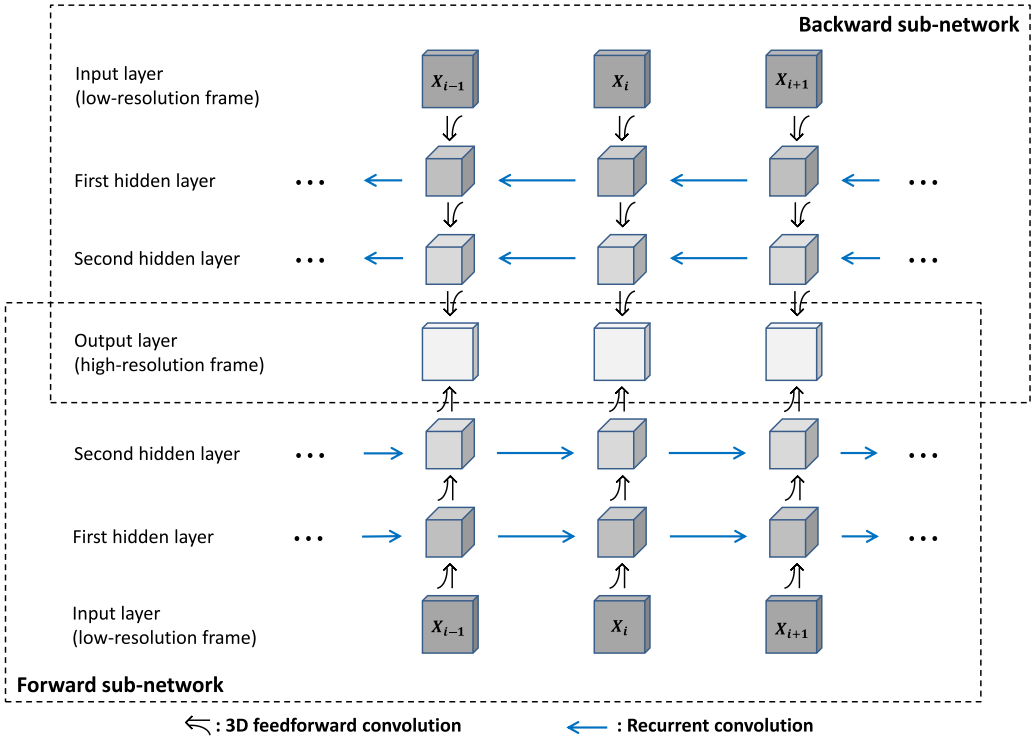


Fig. 1. The proposed bidirectional recurrent convolutional network (BRCN).

$$\begin{aligned}\mathbf{H}_{1,i}^{f} &= \lambda(\mathbf{W}_1^f \hat{*} [\mathbf{X}]_i^f + \mathbf{U}_1^f * \mathbf{H}_{1,i-1}^f + \mathbf{B}_1^f) \\ \mathbf{H}_{1,i}^{b} &= \lambda(\mathbf{W}_1^b \hat{*} [\mathbf{X}]_i^b + \mathbf{U}_1^b * \mathbf{H}_{1,i-1}^b + \mathbf{B}_1^b),\end{aligned} \tag{1}$$

$$h_j^{xyz} = \sum_{k=0}^{c-1} \sum_{h=0}^{s_{w_1}-1} \sum_{w=0}^{s_{w_1}-1} \sum_{t=0}^{t_{w_1}-1} w_{kj}^{hwt} x_k^{x+h,y+w,z+t}, \tag{2}$$

where $\mathbf{U}_1^f$ (or $\mathbf{U}_1^b$) represents the filters of recurrent convolution in the forward (or backward) sub-network. The size is $n_1 \times s_{u_1} \times s_{u_1} \times n_1$, where $s_{u_1} \times s_{u_1}$ is the filter size and n_1 is the number of filters, as well as the number of feature maps in $\mathbf{H}_{1,i}^f$ (or $\mathbf{H}_{1,i}^b$). $\mathbf{B}_1^f$ (or $\mathbf{B}_1^b$) represents biases. $*$ denotes the conventional 2D convolution. The activation function is the rectified linear unit (ReLu): $\lambda(x) = \max(0, x)$ [35].

$\mathbf{W}_1^f$ and $\mathbf{W}_1^b$ represent the filters of 3D feedforward convolution in the forward and backward sub-networks, respectively. Both of them have the size of $c \times s_{w_1} \times s_{w_1} \times t_{w_1} \times n_1$, where c is the number of input channels of a video frame, $s_{w_1} \times s_{w_1}$ is the spatial filter size, and t_{w_1} is the temporal step. $\hat{*}$ denotes the 3D convolution which can be formulated as follows:

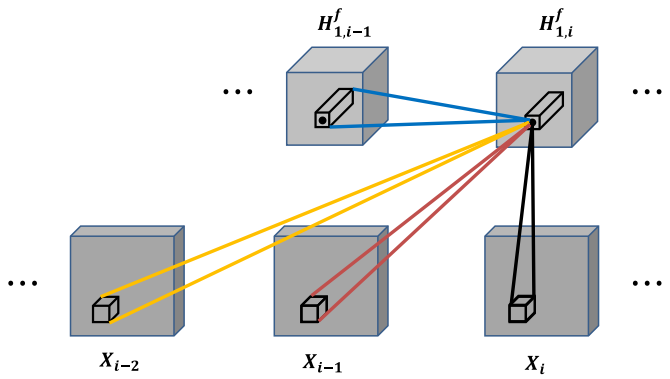


Fig. 2. The details of 3D feedforward and recurrent convolutions (best viewed in colors).

where h_j^{xyz} is the value at position (x, y, z) on the jth feature map in $\mathbf{H}_{1,i}^f$, $x_k^{x+h,y+w,z+t}$ is the value at position $(x+h, y+w, z+t)$ on the kth stacked input channel in $[\mathbf{X}]_i^f$, and w_{kj}^{hwt} is the value at position (h, w, t) in the filter connected between the kth stacked input channel and jth feature map.

Note that conventional 2D feedforward convolution can feed only one frame at the current timestep into the hidden layer. It focuses on the representation learning of spatial content for the current frame. In contrast, our 3D convolution treats stacked adjacent frames as input, and additionally handles motion information by making comparison among the current and its previous frames. Similar 3D convolution has been previously explored in [26], [47], but differs from them, ours is bidirectional in the context of RNN rather than undirected in CNN. Particularly, in the forward (or backward) sub-network, the convolution takes ith frame and only its previous (or future) frames as inputs, rather than all the frames in their cases. In our network, the 3D feedforward convolution also cooperates with directional recurrent convolution to jointly model the temporal dependency in video frames.

Second Hidden Layer. This phase projects the obtained feature maps $\mathbf{H}_{1,i}^f$ (or $\mathbf{H}_{1,i}^b$) to another hidden layer, which aims to capture the nonlinear structure in the video sequence. In addition to intra-frame mapping by conventional feedforward convolution [9], we consider two inter-frame mappings using recurrent and 3D feedforward convolutions, respectively. The projected feature maps in the second hidden layer $\mathbf{H}_{2,i}^f$ (or $\mathbf{H}_{2,i}^b$) in the forward (or backward) sub-network can be obtained as follows:

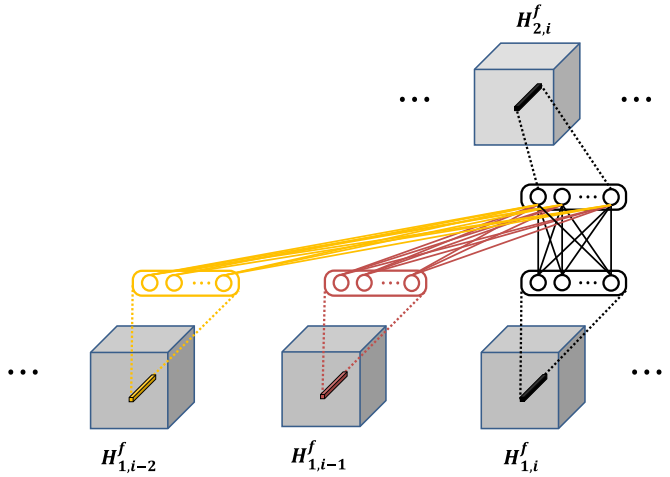


Fig. 3. 3D feedforward convolution as local full connection (filter size: 1×1×3).

$$\begin{aligned}\mathbf{H}^f_{2,i} &= \lambda(\mathbf{W}^f_2 \hat{*} [\mathbf{H}]^f_{1,i} + \mathbf{U}^f_2 * \mathbf{H}^f_{2,i-1} + \mathbf{B}^f_2)\\ \mathbf{H}^b_{2,i} &= \lambda(\mathbf{W}^b_2 \hat{*} [\mathbf{H}]^b_{1,i} + \mathbf{U}^b_2 * \mathbf{H}^b_{2,i-1} + \mathbf{B}^b_2),\end{aligned} \tag{3}$$

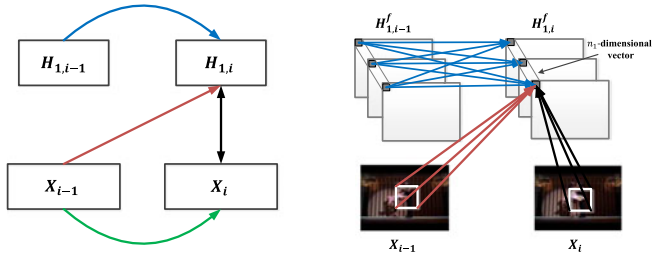
where $\mathbf{W}^f_2$ and $\mathbf{W}^b_2$ represent the filters of 3D feedforward convolutions in the forward and backward sub-networks, respectively. Both of them have the size of $n_1 \times s_{w_2} \times s_{w_2} \times t_{w_2} \times n_2$. $\mathbf{U}^f_2$ and $\mathbf{U}^b_2$ represent the filters of recurrent convolution, their sizes are $n_2 \times s_{u_2} \times s_{u_2} \times n_2$.

Note that the goal of modeling nonlinear structure is achieved by fixing the filter size $s_{w_2} \times s_{w_2}$ of 3D feedforward convolution to be 1×1. So this convolution is equivalent to a local full connection with nonlinear activation function. Different from traditional full connection, this one occurs locally instead of globally that aims to detect more subtle nonlinear details. Particularly, it temporally fuses multiple feature vectors at each spatial location within different timesteps, and efficiently slides over all the locations of feature maps in a similar way as convolution. As shown in Fig. 3, the elements at the same spatial location across n_1 feature maps in $\mathbf{H}^f_{1,i}$ (or $\mathbf{H}^f_{1,i-1}$, $\mathbf{H}^f_{1,i-2}$) comprise a n_1-dimensional vector. These vectors are then concatenated and fully-connected to a n_2-dimensional vector obtained by n_2 feature maps in $\mathbf{H}^f_{2,i}$. Similarly, such fully-connected scheme is shared among all the other locations in a sliding manner. This kind of convolution can also be regarded as a 3D version of "network in network" [31], where the nonlinear projection is performed in both spatial and temporal dimensions rather than spatial only.

It should be noted that the inference of the two hidden layers can be regarded as a representation learning phase. We could stack more hidden layers to increase the representability of our network. But it will dramatically increase the network complexity, so we only use two hidden layers.

Output Layer. When super resolving a low-resolution frame, its visual content is related to not only its adjacent frames at previous timesteps but also the frames at future timesteps. So we use two directional sub-networks to separately model these temporal dependencies along the time axis and reverse, respectively. We combine their obtained n_2-dimensional feature maps in second hidden layers to jointly predict the desired high-resolution frame

$$\mathbf{O}_i = \mathbf{W}^f_3 \hat{*} [\mathbf{H}]^f_{2,i} + \mathbf{W}^b_3 \hat{*} [\mathbf{H}]^b_{2,i} + \mathbf{B}^f_3 + \mathbf{B}^b_3, \tag{4}$$



(a) TRBM [41] (b) BRCN

Fig. 4. Comparison between TRBM and the proposed BRCN. For both models, we just show details at two timesteps for clear illustration.

where $\mathbf{W}^f_3$ and $\mathbf{W}^b_3$ represent the filters of 3D feedforward convolutions in forward and backward sub-networks, respectively. Their sizes are both $n_2 \times s_{w_3} \times s_{w_3} \times t_{w_3} \times c$. We do not use any recurrent convolution for output layer.

3.2 Connection with Temporal Restricted Boltzmann Machine

We will discuss the connection between the proposed BRCN and temporal restricted boltzmann machine (TRBM) [43] that has similar network architecture, with the goal to illustrate that BRCN can model the long-term contextual information in a finer level, i.e., patch-based rather than frame-based.

As shown in Fig. 4, TRBM and BRCN contain similar recurrent connections (blue lines) between pairwise hidden layers, and multi-step feedforward connections (red and black lines) between input layers and hidden layer. They share the common flexibility to model and propagate temporal dependency along the time. However, TRBM is a generative model while BRCN is a discriminative model, and TRBM contains an additional connection (green line) between pairwise input layers for data generation.

In fact, BRCN can be regarded as a deterministic, bidirectional and patch-based implementation of TRBM. Specifically, when inferring the hidden layer in BRCN, as illustrated in Fig. 4 (b), 3D feedforward convolution extracts temporally overlapped patches from the inputs, each of which is fully connected to a n_1-dimensional vector in the feature maps $\mathbf{H}^f_{1,i}$. For recurrent convolution, since each filter size is 1×1^2 and all the filters contain $n_1 \times n_1$ weights, a n_1-dimensional vector in $\mathbf{H}^f_{1,i}$ is fully connected to the corresponding n_1-dimensional vector in $\mathbf{H}^f_{1,i-1}$ at the previous timestep. Therefore, the patch connections of BRCN are actually those of a "discriminative" TRBM. In other words, by setting the spatial filter size of 3D feedforward convolution as the size of the whole frame, BRCN is equivalent to TRBM.

Compared with TRBM, BRCN has the following advantages for handling the task of video SR. 1) BRCN restricts the receptive field of original full connection to a patch rather than the whole frame, which can capture the temporal change of visual details. 2) BRCN replaces all the full connections with weight-sharing convolutional ones, which largely reduces the computational cost. 3) BRCN is more flexible to handle videos of different sizes, once it is trained on a fixed-size video dataset. Similar to TRBM, the proposed model can be generalized to other sequence modeling applications, e.g., video motion modeling [45].

2. We select 1×1 as our default filter size of recurrent convolution as explained in Section 4.2.1.

TABLE 1
The Results of PSNR (dB) and Time (sec) by the Proposed Model with Different Filter Sizes of Recurrent Convolution (Filter Size) and Temporal Lengths of Training Volume (Length)

Filter size	Length	*Dancing*		*Flag*		*Fan*		*Treadmill*		*Turbine*		**Average**	
		PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
1×1	5	28.03	3.19	28.43	0.76	33.74	1.46	22.68	0.46	27.54	0.60	28.08	1.29
	10	28.01		28.46		33.75		22.68		27.54		28.08	
	15	28.02		28.50		33.75		22.66		27.52		28.09	
3×3	5	28.02	9.41	28.43	2.24	33.72	4.13	22.66	1.34	27.37	1.77	28.04	3.78
	10	28.03		28.45		33.69		22.65		27.42		28.05	
	15	28.06		28.44		33.68		22.67		27.45		28.06	
5×5	5	28.00	19.88	28.39	4.76	33.67	8.76	22.61	2.87	27.31	3.76	27.99	8.01
	10	27.95		28.41		33.71		22.65		27.35		28.01	
	15	27.98		28.52		33.75		22.59		27.29		28.03	

3.3 Network Learning

Through combining Equations (1), (3) and (4), we can obtain the desired prediction $\mathbf{O}(\mathcal{X};\Theta)$ from the low-resolution video $\mathcal{X}$, where Θ denotes the network parameters. Network learning proceeds by minimizing the mean square error (MSE) between the predicted high-resolution video $\mathbf{O}(\mathcal{X};\Theta)$ and the groundtruth $\mathcal{Y}$

$$\mathcal{L} = \|\mathbf{O}(\mathcal{X};\Theta) - \mathcal{Y}\|^2, \tag{5}$$

via stochastic gradient descent. Actually, stochastic gradient descent is enough to achieve satisfying results, although we could exploit other advanced optimization algorithms but with higher computational cost, e.g., L-BFGS.

During optimization, the filter weights of 3D feedforward convolution at the $1\mathrm{st}$ temporal step are pre-trained on static images [9], whereas the filter weights of recurrent and 3D feedforward convolutions at the rest temporal steps are initialized by randomly sampling from a Gaussian distribution with mean 0 and standard deviation 0.001. Note that the pretraining step only aims to speed up training by providing a better parameter initialization, due to the limited size of training set. This step could be avoided by alternatively using a larger scale training set. We experimentally find that using a smaller learning rate (e.g., $1e-4$) for the weights in the output layer is crucial to obtain good performance.

4 Experimental Results

To verify the effectiveness, we apply the proposed model to the task of video SR, and present both quantitative and qualitative results as follows.

4.1 Datasets and Implementation Details

We use 25 YUV format video sequences[3] as our training set, which have been widely used in many video SR methods [32], [37], [44]. To enlarge the training set, model training is performed in a volume-based way, i.e., cropping multiple overlapped volumes from training videos and then regarding each volume as a training sample. During cropping, each volume has a spatial size of 32×32 and a temporal length of 10. The spatial and temporal strides are 14 and 8, respectively. As a result, we can generate roughly 41,000 volumes from the original dataset.

We test our model on two datasets, namely Set1 and Set2. Set1 includes a variety of challenging videos: *Dancing*, *Flag*, *Fan*, *Treadmill* and *Turbine* [40], which contain complex motions with severe motion blur and aliasing. Set2 has 4 videos: *City*, *Calendar*, *Foliage* and *Walk*, which have been used by state-of-the-art methods [30], [32]. Note that we do not have to extract volumes during testing, since the convolutional operation can scale to videos of any spatial size and temporal length.

We generate the corresponding low-resolution videos for both datasets with the following steps: 1) using Gaussian filter with standard deviation 2 to smooth each original frame, and 2) downsampling the frame by a factor with bicubic method.[4]

4.2 Network Architecture

Some important parameters of our proposed network are illustrated as follows: s_{w_1} =9, t_{w_1} =3, s_{w_2} =1, t_{w_2} =3, s_{w_3} =5, t_{w_3} =3, s_{u_1} =1, s_{u_2} =1, n_1 =64, n_2 =32, and c =1.[5] Note that for 3D feedforward convolution, varying the number and spatial size of its filters does not have a significant impact on the performance, because some filters with certain spatial sizes are already in a regime where they can already well model the frame content [9], [51]. We will focus on the selections of other parameters especially associated with sequence modeling, and study the relation between performance and parameters, as well as the best trade-off between performance and speed. The evaluation measures include both peak signal-to-noise ratio (PSNR) and testing time (Time).

4.2.1 Filter Size of Recurrent Convolution

We test three different filter sizes of recurrent convolution including 1×1, 3×3 and 5×5. For simple comparison, we experiment with only the forward sub-network of our proposed BRCN on the Set1 dataset with an upscaling factor of 4, as shown in Table 1. We can see that when the temporal length of training volume (Length) is fixed, the larger filter size produces the worse PSNR performance. It is mainly

3. https://media.xiph.org/video/derf/.

4. Here we focus on two factors of 4 and 2 that are usually considered settings in video SR.

5. Similar to [46], we only deal with luminance channel in the YCbCr color space.

TABLE 2
The Results of PSNR (dB) and Time (sec) by the Proposed Model with Different Directions of Network (Direction), Model Sizes (Size), and Temporal Steps of 3D Feedforward Convolution (Step)

Direction	Size	Step	*Dancing*		*Flag*		*Fan*		*Treadmill*		*Turbine*		**Average**	
			PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time	PSNR	Time
U	≈8 k	1	27.81	1.41	28.04	0.36	33.61	0.60	22.42	0.15	27.50	0.23	27.87	0.55
U×2	≈20 k	1	27.92	3.27	28.07	0.77	33.64	1.41	22.46	0.45	27.56	0.59	27.93	1.30
U×4	≈57 k	1	27.94	10.55	28.18	2.45	33.64	4.52	22.47	1.45	27.58	1.92	27.96	4.18
F *w/o f*	≈13 k	1	27.94	2.42	28.30	0.55	33.47	1.03	22.64	0.33	27.49	0.43	27.97	0.95
F *w/o r*	≈16 k	2	28.00	2.11	28.31	0.48	33.21	0.89	22.62	0.28	27.35	0.38	27.90	0.83
F	≈21 k	2	28.01	3.19	28.46	0.76	33.75	1.46	22.68	0.46	27.54	0.60	28.08	1.29
F	≈29 k	3	28.03	4.70	28.53	1.07	33.78	2.03	22.68	0.64	27.56	0.84	28.12	1.85
F	≈37 k	4	27.99	6.07	28.49	1.38	33.65	2.63	22.58	0.83	27.37	1.09	28.02	2.40
B *w/o f*	≈13 k	1	27.98	2.41	28.28	0.55	33.46	1.02	22.62	0.33	27.47	0.43	27.96	0.95
B *w/o r*	≈16 k	2	28.01	2.09	28.26	0.48	33.28	0.90	22.60	0.29	27.35	0.38	27.90	0.83
B	≈21 k	2	28.04	3.17	28.48	0.76	33.76	1.45	22.67	0.46	27.53	0.60	28.09	1.28
B	≈29 k	3	28.07	4.70	28.51	1.06	33.77	2.04	22.68	0.65	27.59	0.84	28.12	1.85
B	≈37 k	4	27.99	6.06	28.50	1.38	33.72	2.63	22.63	0.83	27.54	1.06	28.07	2.39
F+B	≈42 k	2	28.06	3.20	28.50	0.78	33.78	1.47	22.69	0.47	27.61	0.62	28.12	1.30
F+B	≈58 k	3	28.16	4.71	28.57	1.08	33.81	2.06	22.72	0.65	27.73	0.85	28.20	1.87
F+B	≈74 k	4	28.04	6.08	28.52	1.39	33.74	2.65	22.66	0.84	27.55	1.10	28.10	2.41

attributed to the border effects. In particular, when the filter size is larger than 1×1, the size of feature maps in hidden layers will gradually shrink to zero after doing recurrent convolution for multiple timesteps. So we have to perform zero-padding on the shrunk feature maps to keep their sizes from decreasing. In this way, zero-valued pixels are introduced to the border of feature maps, and will degenerate the final prediction of the corresponding positions in high-resolution frames. We can also observe from the table that a larger filter size requires much more testing time, and thus lowers the efficiency of our model. So we use 1×1 as our default filter size of recurrent convolution.

4.2.2 Temporal Length of Training Volume

As stated in Section 4.1, we use volumes as our training data with a temporal length of 10. It means that when super resolving a video frame, we can only use its previous or future frames with a window size of 10 as contextual information. We also experiment with another two temporal lengths in terms of 5 and 15 as presented in Table 1. When the filter size of recurrent convolution (Filter size) is fixed, the longer length achieves slightly better PSNR performance. Ideally, we could use volumes with very long length (>30) for training so that much more contextual information can be used. But RNN-based methods cannot model such long-term dependency relation well due to the problems of gradient vanishing and explosion [4]. Although we could replace the used ReLu activation function with long short-term memory (LSTM) to release these problems, it will greatly increase the complexity of our network by introducing a large number of learning parameters. In addition, even though the testing time of using different temporal lengths is the same, volumes with a longer length will need relatively higher computational cost during training. Therefore, we use ReLu as our activation function and set the temporal length of training volume as 10.

4.2.3 Direction of Network

To demonstrate the effectiveness of our bidirectional scheme in BRCN, we evaluate four networks with different directions as follows:

1) Undirected (U): this network has no recurrent convolution, and its temporal step of 3D feedforward convolution is fixed as 1 similar to a conventional 2D convolution. U×2 (or U×4) enlarges the model size by using two times (or four times) the number of filters per layer in U.
2) Forward (F): forward sub-network which considers the temporal dependency only in the forward direction. F *w/o f* removes the 3D feedforward convolution, while F *w/o r* removes the recurrent convolution.
3) Backward (B): back sub-network which considers the temporal dependency only in the backward direction.
4) Bidirectional (F+B): bidirectional recurrent convolutional network which combines the forward and backward sub-networks together.

The comparison of these networks is illustrated in Table 2. From the table we can see that, without recurrent or 3D feedforward connections for temporal dependency modeling, undirected network performs the worst. Forward and backward sub-networks obtain similar better performance which verifies the usefulness of exploiting contextual information in either forward or backward direction. When combining them together, the bidirectional network can achieve better results. It is attributed to the fact that each video frame is related to not only its previous frames but also the future ones. Note that the testing time of bidirectional networks is not much longer than that of others, because we accelerate it by separately proceeding two directional sub-networks in parallel and then combining their predictions.

4.2.4 Comparison of Two Convolutions

Our proposed model contains two different components for aggregating temporal information of video frames: 3D feedforward convolution and recurrent convolution. To properly disentangle their contributions to the final performance, we report the performance of two kinds of models in Table 2: 1) models using only 3D feedforward convolution but without recurrent convolution (*w/o r*), and 2) models using only

recurrent convolution but without 3D feedforward convolution (*w/o f*). In the forward sub-network, by comparing U with F *w/o r* (or F *w/o f*), we can conclude that only using 3D feedforward convolution (or recurrent convolution) can improve the performance by 0.03-0.10dB. Similar observations can also be found in the backward sub-network. When jointly using the two convolutions as shown in F, the performance can be further improved by 0.11-0.18dB. It indicates that the two convolutions can cooperatively model the temporal dependency in a comprehensive way due to their complementary characteristics.

4.2.5 Temporal Step of 3D Feedforward Convolution

Since there exist a large number of fast-varying motions in video frames, we exploit 3D feedforward convolutions in the proposed network to learn the spatio-temporal patterns. To investigate what is the optimal temporal step that can best describe these motions, we set the temporal step as 2, 3 and 4, respectively, and compare their performance with different model configurations in Table 2. We can find that when the temporal step is 3, we can achieve the best performance for both forward and backward sub-networks, as well as their combinations. It means that the previous or future two frames can provide most helpful contextual information for super resolving the current frame. When the temporal step is larger than 3, the performance slightly reduces. It seems reasonable since most fast-varying motions often occur locally in several frames rather than remain for a long time. But it should be noted that the selection of temporal step also depends on the velocity of motion, the frame rate of video and the frequency of frame sampling.

4.2.6 Model Size

For many deep learning methods, their performance tends to increase with model size, i.e., the number of learnable parameters. For our proposed model, its performance gradually improves when the model size increases from 8 to 58 k. To study whether the performance gains come from the proposed network components or the increased model size, we report the model size of each model configuration in Table 2, and compare our proposed model with three undirected networks with comparable model sizes. When comparing with U$\times$2 (Size$\approx$20 k), the proposed F (Size$\approx$21 k) and B (Size$\approx$21 k) can outperform it by 0.15 and 0.16 dB, respectively. The proposed F+B (Size$\approx$58 k) performs better than U$\times$4 (Size$\approx$57 k) by 0.24 dB with faster speed. These evidences demonstrate the effectiveness of our network components.

4.3 Comparison with State-of-the-Art Methods

We select the network architecture with the best performance as our default BRCN, and compare with seven single-image SR and four multi-frame SR methods including:

- Bicubic: bicubic interpolation.
- SC: sparse coding-based method [52].
- K-SVD: dictionary learning method based on singular value decomposition [53].
- NE+NNLS: nonnegative neighbor embedding [5].
- ANR: anchored neighbor regression method [46].
- NE+LLE: locally linear neighbor embedding [7].
- SR-CNN: image SR with CNN [9].
- 3DSKR: 3D steering kernel regression [44].
- Enhancer: a commercial software for video SR [1].
- FUS: fast video upsampling [41].
- DeepSR: SR draft ensemble learning with CNN [30].

For all these methods, we use their publicly available codes to perform all the experiments.[6] It should be noted that we do not make comparison with the state-of-the-art multi-frame SR method [32], because we do not have the corresponding code and its released results on the Set2 dataset are obtained in a different setting as ours. But instead we compare with the recent state-of-the-art method named DeepSR [30] which is shown to outperform [32] in most cases.

4.3.1 PSNR and SSIM Comparison

The results of all the methods on the Set1 and Set2 datasets are compared in Tables 3 and 4, respectively. In each table, we present the results with two upscaling factors in terms of 2 (Scale2) and 4 (Scale4), and use three evaluation measures including peak signal-to-noise ratio, structural similarity (SSIM) and testing time (Time).

For the upscaling factor of 4, we can see that our BRCN clearly outperforms all the compared methods in both averaged PSNR and SSIM. Specifically, compared with the state-of-the-art single-image SR methods (e.g., SR-CNN, ANR and K-SVD), our multi-frame-based method can surpass them by 0.33$\sim$0.59 dB and 0.30$\sim$0.45 dB in PSNR on the two datasets, respectively. It is mainly attributed to the beneficial mechanism of temporal dependency modeling. BRCN also performs much better than the two representative multi-frame SR methods (3DSKR and Enhancer) on the two datasets. In fact, most existing multi-frame-based methods tend to fail catastrophically when dealing with very complex motions, since it is difficult for them to estimate these motions with pinpoint accuracy. Alternatively, our method exploits recurrent and 3D feedforward convolutions that are able to learn the both long-term content relation and short-term spatio-temporal patterns for challenging motions. Note that overall our BRCN performs slightly worse than the state-of-the-art method DeepSR that additionally exploits multiple SR drafts by different motion estimations with ensemble learning. Different from it, BRCN does not use any time-consuming motion estimation so that it runs much faster while still achieving comparable performance.

For the upscaling factor of 2, we can observe from both tables that our method can outperform other methods by a much larger margin. It demonstrates that our method is better at super resolving the videos with a smaller upscaling factor. In particular, video frames with a smaller upscaling factor usually have more detailed information in their visual content, which can greatly facilitate the learning of local spatio-temporal features for fast-varying motions.

4.3.2 Time versus PSNR

We also present the comparison of Time in both Tables 3 and 4, where all the methods are implemented on the same machine (Intel CPU 3.10 GHz and 32 GB memory). The

6. We retrained SR-CNN on the new dataset, and found that the pre-trained and retrained SR-CNNs obtain similar results.

TABLE 3
The Results of PSNR, SSIM and Time by All the Methods on the Set1 Dataset with Two Upscaling Factors of 2 and 4

Model	*Dancing*		*Flag*		*Fan*		*Treadmill*		*Turbine*		**Average**	
	Scale2	Scale4	Scale2	Scale4	Scale2	Scale4	Scale2	Scale4	Scale2	Scale4	Scale2	Scale4
PSNR												
Bicubic	29.11	26.83	28.95	26.35	34.34	31.94	23.68	21.15	28.08	25.09	28.83	26.27
SC [52]	30.04	26.80	23.89	26.28	26.43	32.50	22.12	21.27	21.87	25.77	24.87	26.52
K-SVD [53]	29.88	27.69	29.91	27.61	34.79	33.55	24.37	22.22	29.04	27.00	29.59	27.61
NE+NNLS [5]	29.72	27.63	29.69	27.41	34.74	33.45	24.28	22.08	28.91	26.88	29.46	27.49
ANR [46]	29.81	27.67	29.70	27.52	34.71	33.49	24.33	22.24	28.94	27.04	29.49	27.59
NE+LLE [7]	29.78	27.64	29.70	27.48	34.69	33.46	24.33	22.22	28.93	26.98	29.48	27.52
SR-CNN [9]	30.00	27.81	30.01	28.04	34.75	33.61	24.41	22.42	29.27	27.50	29.69	27.87
3DSKR [44]	29.51	27.81	29.72	26.89	34.50	31.91	24.38	22.32	28.20	24.27	29.17	26.64
Enhancer [1]	29.40	27.06	28.99	26.58	34.86	32.14	24.27	21.20	28.37	25.60	29.18	26.52
BRCN	34.36	28.16	33.44	28.57	36.20	33.81	26.87	22.72	33.35	27.73	32.84	28.20
SSIM												
Bicubic	0.8434	0.7124	0.8246	0.7091	0.8575	0.8163	0.7518	0.5978	0.8319	0.7219	0.8218	0.7115
SC [52]	0.5745	0.6010	0.8383	0.7061	0.8731	0.8029	0.7807	0.6030	0.8324	0.7014	0.7798	0.6828
K-SVD [53]	0.8626	0.6699	0.8569	0.7514	0.8666	0.8219	0.7896	0.6448	0.8569	0.7548	0.8465	0.7285
NE+NNLS [5]	0.7355	0.6074	0.8517	0.7387	0.8671	0.8204	0.7841	0.6297	0.8536	0.7458	0.8184	0.7084
ANR [46]	0.8608	0.6166	0.8512	0.7440	0.8676	0.8227	0.7861	0.6430	0.8555	0.7579	0.8442	0.7168
NE+LLE [7]	0.7032	0.6231	0.8507	0.7429	0.8645	0.8212	0.7834	0.6422	0.8537	0.7565	0.8111	0.7171
SR-CNN [9]	0.7889	0.7966	0.8588	0.7597	0.8659	0.8270	0.7943	0.6689	0.8604	0.7672	0.8336	0.7638
3DSKR [44]	0.7530	0.6521	0.8530	0.7420	0.8501	0.8120	0.7723	0.6450	0.8430	0.7263	0.8142	0.7154
Enhancer [1]	0.9094	0.7364	0.8359	0.7286	0.8495	0.7901	0.7715	0.6365	0.8433	0.7306	0.8419	0.7245
BRCN	0.8541	0.7757	0.9080	0.7764	0.9325	0.8499	0.8661	0.6818	0.9238	0.7860	0.8968	0.7739
Time												
Bicubic	-	-	-	-	-	-	-	-	-	-	-	-
SC [52]	109.42	45.47	34.80	12.89	28.26	12.92	41.36	15.47	39.17	16.49	50.60	20.64
K-SVD [53]	7.17	2.35	1.83	0.58	3.19	1.06	1.15	0.35	1.63	0.51	2.99	0.97
NE+NNLS [5]	71.47	19.89	18.84	4.54	33.05	8.27	11.46	2.60	15.58	3.67	30.08	7.79
ANR [46]	1.98	0.85	0.49	0.20	0.79	0.38	0.31	0.12	0.44	0.18	0.80	0.35
NE+LLE [7]	13.61	4.20	3.39	0.96	5.61	1.76	2.05	0.57	2.82	0.80	5.49	1.66
SR-CNN [9]	1.42	1.41	0.37	0.36	0.63	0.60	0.16	0.15	0.25	0.23	0.56	0.55
3DSKR [44]	4224	1211	957	255	1580	323	443	127	697	173	1580	418
Enhancer [1]	-	-	-	-	-	-	-	-	-	-	-	-
BRCN	4.70	4.71	1.08	1.08	2.05	2.06	0.65	0.65	0.85	0.85	1.87	1.87

publicly available codes of compared methods are all in MATLAB while SR-CNN and ours are in Python. For clear illustration, we plot Time versus PSNR on the Set2 dataset in Fig. 7. From the figure, we can see that our BRCN takes 10.27 sec per frame on average, which is orders of magnitude faster than the fastest multi-frame SR method 3DSKR. It should be noted that the speed gap is not caused by the different MATLAB/Python implementations. As stated in [32], [44], the computational bottleneck for the existing multi-frame SR methods is the accurate motion estimation, while our model explores an alternative based on efficient convolutions which has much lower computational complexity. Note that the speed of our method is worse than the fastest single-image SR method ANR. It is likely that our method involves the additional phase of temporal dependency modeling, so that we can achieve better performance. In addition, compared with recent state-of-the-art single-image SR methods including [28], [29], [42], some of them might achieve better performance than our model, but they use a much deeper network with more learnable parameters, so they should be much slower than our BRCN. We also test our model on a NVIDIA K20 GPU (denoted as BRCN-GPU) and achieve real-time video SR by taking 0.48 sec per frame.

4.3.3 Visual Comparison

In addition to the quantitative evaluation, we present some qualitative results. Since we only deal with the luminance channel in the YCbCr color space, here we simply upscale the other two channels with bicubic interpolation for well illustration [46]. We present the closeup comparison with the best performed methods on the two datasets with different upscaling factors in Figs. 5 and 6, respectively. *Please enlarge and view these figures on the screen for better comparison*. From these figures, we can observe that our method is able to recover more image details than others under various motion conditions.

4.4 Color Video Super-Resolution

In previous experiments, we deal with color videos by first transforming video frames from RGB space to YCbCr space and then dealing with only luminance channel in the YCbCr space. Next we will consider to simultaneously handle all three channels in the RGB space. We can achieve this goal by resetting the number of input channels as $c = 3$, and feeding more channels into the proposed network without changing the learning procedure and network architecture.

We perform experiments of color video SR on the Set2 dataset. The comparison methods contain a baseline

TABLE 4
The Results of PSNR, SSIM and Time by All the Methods on the Set2 Dataset with Two Upscaling Factors of 2 and 4

Model	*City*		*Calendar*		*Foliage*		*Walk*		**Average**	
	Scale2	Scale4	Scale2	Scale4	Scale2	Scale4	Scale2	Scale4	Scale2	Scale4
						PSNR				
Bicubic	26.26	24.81	21.54	20.17	24.97	23.00	27.89	25.35	25.16	23.30
SC [52]	25.25	24.70	22.20	20.17	24.82	22.92	26.78	25.17	24.76	23.24
K-SVD [53]	26.75	25.24	22.11	20.71	25.59	23.62	28.74	26.38	25.80	23.98
NE+NNLS [5]	26.68	25.18	22.05	20.64	25.52	23.54	28.61	26.22	25.72	23.89
ANR [46]	26.74	25.26	22.07	20.73	25.61	23.69	28.71	26.45	25.78	24.03
NE+LLE [7]	26.73	25.24	22.06	20.69	25.60	23.64	28.68	26.35	25.77	23.98
SR-CNN [9]	26.78	25.21	22.22	20.87	25.66	23.74	28.80	26.69	25.86	24.13
3DSKR [44]	26.50	25.22	22.01	20.70	25.31	23.65	28.75	26.33	25.64	23.97
Enhancer [1]	26.41	25.32	21.93	20.64	25.25	23.64	28.19	26.18	25.45	23.95
*DeepSR [30]	-	25.72	-	21.39	-	24.92	-	26.67	-	24.67
BRCN	29.12	25.44	24.30	21.09	28.86	24.14	32.80	27.09	28.77	24.43
						SSIM				
Bicubic	0.6725	0.5130	0.6572	0.5177	0.6701	0.4892	0.8456	0.7488	0.7114	0.5677
SC [52]	0.6601	0.5057	0.6921	0.5051	0.7043	0.4840	0.8370	0.7368	0.7234	0.5579
K-SVD [53]	0.7099	0.5495	0.6965	0.5528	0.7094	0.5352	0.8681	0.7818	0.7460	0.6048
NE+NNLS [5]	0.7046	0.5428	0.6911	0.5417	0.6991	0.5253	0.8651	0.7683	0.7400	0.5945
ANR [46]	0.7105	0.5517	0.6981	0.5525	0.7117	0.5427	0.8684	0.7828	0.7472	0.6074
NE+LLE [7]	0.7094	0.5501	0.6906	0.5507	0.7102	0.5407	0.8677	0.7799	0.7445	0.6054
SR-CNN [9]	0.7168	0.5535	0.7022	0.5745	0.7154	0.5515	0.8684	0.7881	0.7507	0.6169
3DSKR [44]	0.7095	0.5621	0.6978	0.5733	0.7098	0.5663	0.8678	0.7810	0.7462	0.6206
Enhancer [1]	0.7123	0.5822	0.6920	0.5741	0.7102	0.5656	0.8639	0.7799	0.7446	0.6254
*DeepSR [30]	-	0.6499	-	0.6937	-	0.7250	-	0.7694	-	0.7095
BRCN	0.8175	0.5751	0.7954	0.5878	0.8560	0.5865	0.9307	0.7842	0.8499	0.6334
						Time				
Bicubic	-	-	-	-	-	-	-	-	-	-
SC [52]	461.97	177.49	638.58	243.35	644.27	240.69	370.62	169.89	528.86	207.86
K-SVD [53]	24.35	7.36	24.09	6.95	18.36	5.16	18.42	5.94	21.40	6.35
NE+NNLS [5]	237.40	63.34	225.20	56.93	174.71	40.72	205.99	48.51	210.83	52.38
ANR [46]	5.96	2.76	5.45	2.46	4.48	1.90	4.61	2.16	5.13	2.32
NE+LLE [7]	42.52	13.90	40.88	12.44	30.97	9.05	34.18	10.76	37.14	11.54
SR-CNN [9]	3.44	3.43	3.52	3.51	2.92	2.92	2.92	2.92	3.20	3.20
3DSKR [44]	11929	3276	12050	3343	9935	2739	9935	2739	10964	3024
Enhancer [1]	-	-	-	-	-	-	-	-	-	-
*DeepSR [30]	-	6620	-	6750	-	5239	-	5239	-	5962
BRCN	10.23	10.21	11.17	11.14	9.87	9.87	9.86	9.87	10.28	10.27

* *indicates ensemble method.*

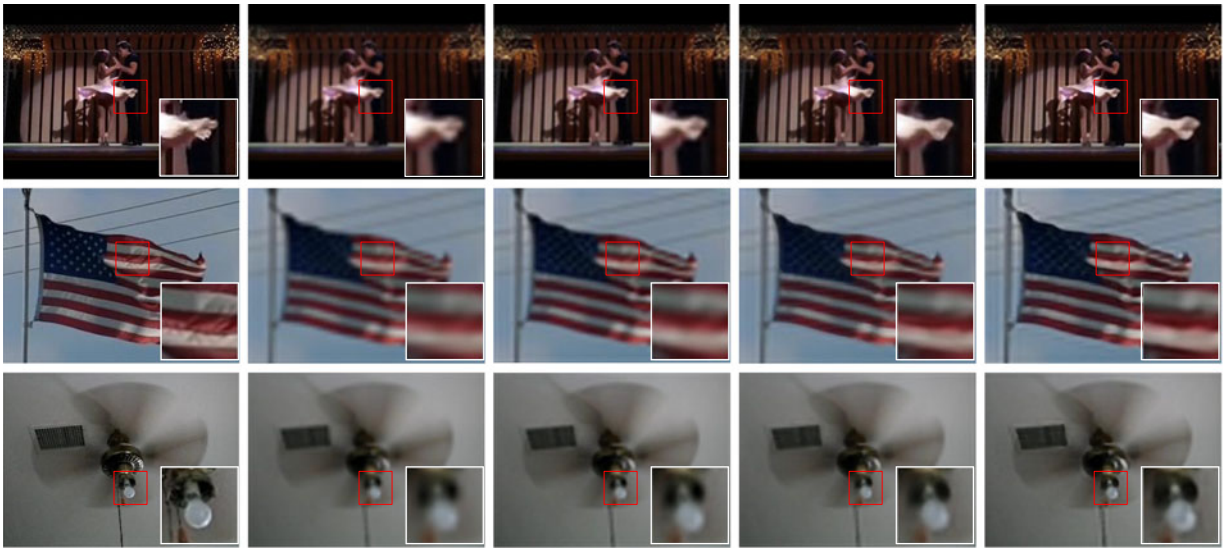
method: Bicubic and three state-of-the-art color video SR methods: Enhancer [1], FUS [41] and DeepSR [30]. As we have demonstrated the effectiveness of BRCN on different upscaling factors, here we only evaluate the performance with the upscaling factor of 4. We also try two different learning strategies for the proposed BRCN: 1) Y only: this is a single-channel (c =1) network trained only on the luminance channel. While Cb and Cr channels are directly upscaled using bicubic interpolation, and 2) RGB: this is a three-channel (c =3) network trained on the RGB channels.

The comparisons of PSNR and SSIM by all the methods are shown in Table 5, where PSNR and SSIM are computed in RGB and gray spaces, respectively. From the table, we can see that our BRCN trained on RGB channels can achieve better performance than the one trained only on the luminance channel. Both two BRCNs can outperform FUS and Enhancer by a large margin. Although without ensemble learning or motion estimation, our BRCN-RGB can still obtain comparable results as DeepSR, which demonstrates its effectiveness on modeling temporal dependency for color videos.

We also present some qualitative results by visually comparing all the super resolved results in Fig. 8. Our BRCN can achieve much clearer results than Enhancer and FUS for all the videos. Although BRCN quantitatively performs worse than DeepSR in Table 5, its super resolved results look more nature and have fewer obtrusive artifacts when large motion occurs, e.g., the running car in the *Foliage* video and the moving leg in the *Walk* video.

4.5 Filter and Feature Map Visualization

To study whether the proposed BRCN can learn meaningful spatio-temporal patterns, we visualize the learned filters of 3D feedforward convolutions with upscaling factors of both 4 and 2. For each upscaling factor, we show the filters in both the first hidden layer and output layer in the forward sub-network, denoted by $\mathbf{W}_1^f$ and $\mathbf{W}_3^f$, respectively. Since $\mathbf{W}_1^f$ and $\mathbf{W}_3^f$ are both 3D filters, to obtain a clear visualization, we set the temporal step of $\mathbf{W}_1^f$ and $\mathbf{W}_3^f$ as 3, and sequentially present the detailed 2D filter weights at each temporal step.



(a) Original (b) Bicubic (c) ANR [44] (d) SR-CNN [9] (e) BRCN

Fig. 5. Closeup comparison among original frames and super resolved results by Bicubic, ANR, SR-CNN and BRCN, respectively, on the Set1 dataset with an upscaling factor of 4.

(a) Original (b) Bicubic (c) ANR [44] (d) SR-CNN [9] (e) BRCN

Fig. 6. Closeup comparison among original frames and super resolved results by Bicubic, ANR, SR-CNN and BRCN, respectively, on the Set2 dataset with an upscaling factor of 2.

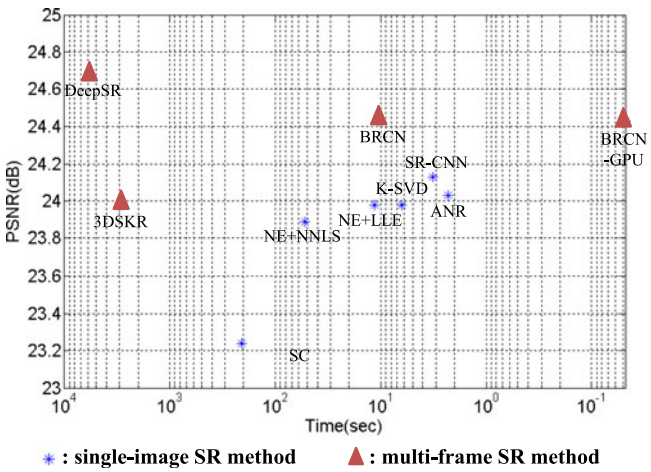


Fig. 7. Time versus PSNR for all the methods.

The learned filters for upscaling factors 4 and 2 are illustrated in Figs. 9 and 10, respectively. The filters of $\mathbf{W}_1^f$ at 1st temporal step exhibit some strip-like patterns, which can be viewed as edge detectors. The filters at 2nd and 3rd temporal steps show similar patterns but in a smoother style, and their phases vary along with temporal steps. It indicates that when super resolving a video frame, not only its current visual content is detected but also the relation to the contextual information in its adjacent frames is temporally modelled.

The filters of $\mathbf{W}_3^f$ at 1st temporal step show some centrally-averaging patterns, which illustrates that the predicted high-resolution frame is obtained by averaging over the feature maps in the second hidden layer. This averaging operation is also in consistent with the corresponding reconstruction phase in patch-based SR methods (e.g., [52]), but the difference is that our filters are automatically learned rather than pre-defined. Regarding the learned filters at 2nd

TABLE 5
The Results of PSNR and SSIM for Color Video SR on the Set2 Dataset with an Upscaling Factor of 4

Model	*City*	*Calendar*	*Foliage*	*Walk*	**Average**
			PSNR		
Bicubic	21.91	17.26	20.06	21.60	20.21
FUS [41]	21.94	17.35	19.90	21.37	20.14
Enhancer [1]	23.22	19.16	22.29	24.82	22.37
*DeepSR [30]	24.24	19.86	23.51	25.26	23.22
BRCN-Y only	23.31	19.57	22.72	25.82	22.85
BRCN-RGB	23.53	19.68	23.03	25.58	22.96
			SSIM		
Bicubic	0.4220	0.4699	0.4275	0.6846	0.5010
FUS [41]	0.4251	0.4870	0.4380	0.6910	0.5103
Enhancer [1]	0.5481	0.5730	0.5702	0.7752	0.6166
*DeepSR [30]	0.6507	0.6938	0.7248	0.7714	0.7102
BRCN-Y only	0.5447	0.5985	0.6018	0.8081	0.6383
BRCN-RGB	0.5685	0.6325	0.6213	0.8247	0.6617

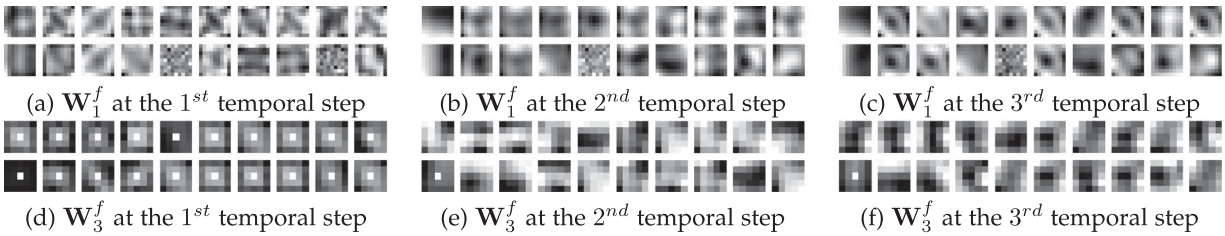
* *indicates ensemble method.*

and 3rd temporal steps, we can observe the clear phase difference between pairwise filters, which demonstrates that our model can learn meaningful time-varying features. When comparing the learned filters of different upscaling factors, we can see that the filters of a smaller factor show more finer-grained patters for describing detailed content, e.g., line-like patterns in Figs. 10b and 10c.

To investigate how such spatio-temporal filters can affect the results of representation learning in the hidden layers, we also show the feature maps of both the first and second hidden layers in Fig. 11. The inputs are 3 successive low-resolution video frames interpolated by the bicubic method in advance. The outputs are predicted high-resolution frames by the proposed BRCN. As we can see, feature maps of the first hidden layer highlight various edges at different directions. By carefully zooming in these feature maps, we can find there exist obvious motion-specific patterns around moving objects, e.g., the dark and white pixels at the top edge of skit (marked by yellow circles in the middle row). It is likely

(a) Input (b) Bicubic (c) FUS [39] (d) Enhancer [1] (e) DeepSR [28] (f) BRCN

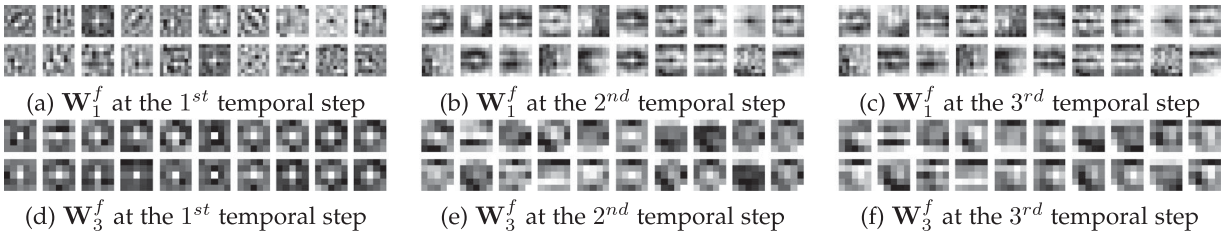
Fig. 8. Visual comparison among input low-resolution frames and super resolved results by Bicubic, FUS, Enhancer, DeepSR and BRCN, respectively, on the Set2 dataset with an upscaling factor of 4.



(a) $\mathbf{W}_1^f$ at the 1^{st} temporal step (b) $\mathbf{W}_1^f$ at the 2^{nd} temporal step (c) $\mathbf{W}_1^f$ at the 3^{rd} temporal step

(d) $\mathbf{W}_3^f$ at the 1^{st} temporal step (e) $\mathbf{W}_3^f$ at the 2^{nd} temporal step (f) $\mathbf{W}_3^f$ at the 3^{rd} temporal step

Fig. 9. Visualization of learned filters in the first hidden and output layers of the proposed BRCN, with an upscaling factor of 4.



(a) $\mathbf{W}_1^f$ at the 1^{st} temporal step (b) $\mathbf{W}_1^f$ at the 2^{nd} temporal step (c) $\mathbf{W}_1^f$ at the 3^{rd} temporal step

(d) $\mathbf{W}_3^f$ at the 1^{st} temporal step (e) $\mathbf{W}_3^f$ at the 2^{nd} temporal step (f) $\mathbf{W}_3^f$ at the 3^{rd} temporal step

Fig. 10. Visualization of learned filters in the first hidden and output layers of the proposed BRCN, with an upscaling factor of 2.

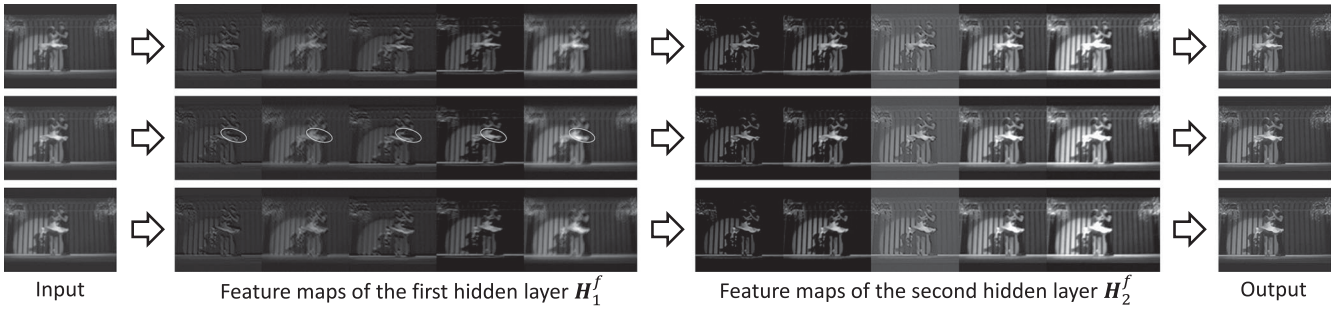


Fig. 11. Visualization of feature maps in the two hidden layers by inputting 3 successive video frames.

that such motion-specific patterns for the current frame are obtained by inferring the moving trend of skit from the previous frames. While for the feature maps of the second hidden layer, they differ mainly on intensities and contain similar motion-specific patterns. It is because the second hidden layer focuses on nonlinearly fusing all the previous feature maps to more steady states for the following final prediction.

5 Conclusion and Future Work

In this paper, we have proposed the bidirectional recurrent convolutional network (BRCN) for multi-frame SR. Our main contribution is the novel use of bidirectional scheme, recurrent and 3D feedforward convolutions for efficient temporal dependency modeling. We have applied our model to super resolve videos containing complex motions, and achieved better performance and faster speed.

Our proposed model has the following limitations: 1) to seek a good balance between performance and efficiency, the model has a very shallow architecture (i.e., only 3 layers), so its performance might not be the best among all the recently released papers [28], [29], [42], 2) the low-resolution images have to be pre-upsampled to the desired size before inputting to the model, thus the computational complexity grows quadratically with the upsampled spatial size, and 3) due to the lack of large-scale video SR dataset, our model is not directly learned from scratch but fine-tuned from pre-trained weights on static images.

To further improve the performance, we will extend our model to have a deeper architecture, e.g., based on 19 layers VGG net [28], or incorporate some effective strategies, e.g., motion ensemble [30]. Although such complex extensions could increase the computational complexity and thus slow the running speed, they will help the model to better model nonlinear properties in image spaces.

For speed acceleration, we will replace the previously used bicubic pre-upsampling by automatically learning diverse upsampling filters with deconvolution layers [11]. In this way, the computational complexity is only proportional to the small spatial size of original low-resolution images, which can thus be largely reduced.

To learn our model with more flexibility, we plan to collect a large-scale high-resolution video dataset, and try to learn our model directly from raw videos.

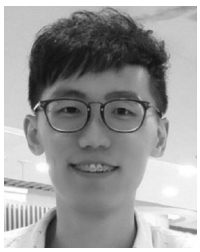
Acknowledgments

This work is jointly supported by National Key Research and Development Program of China (2016YFB1001000), National Natural Science Foundation of China (61525306, 61633021, 61572504, 61420106015), Strategic Priority Research Program of the CAS (XDB02070100), and Beijing Natural Science Foundation (4162058). This work is also supported by grants from NVIDIA and the NVIDIA DGX-1 AI Supercomputer.

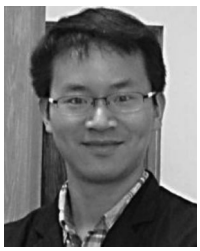
References

[1] Video enhancer. 2014. [Online]. Available: http://www.infognition.com/videoenhancer/, version 1.9.10
[2] S. Baker and T. Kanade, "Super-resolution optical flow," Carnegie Mellon University, Pittsburgh, PA, USA, pp. 99–36, 1999.
[3] B. Bascle, A. Blake, and A. Zisserman, "Motion deblurring and super-resolution from an image sequence," in *Proc. Eur. Conf. Comput. Vis.*, 1996, pp. 571–582.
[4] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[5] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, "Low-complexity single-image super-resolution based on non-negative neighbor embedding," in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 1–10.
[6] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 25–36.
[7] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2004, pp. I–I.
[8] J. Donahue, et al., "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2625–2634.
[9] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 184–199.
[10] C. Dong, C. C. Loy, 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.
[11] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 391–407.
[12] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 633–640.
[13] R. Fransens, C. Strecha, and L. V. Gool, "Optical flow based super-resolution: A probabilistic approach," *Comput. Vis. Image Understanding*, vol. 106, pp. 106–115, 2007.
[14] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *Int. J. Comput. Vis.*, vol. 40, pp. 25–47, 2000.
[15] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 349–356.
[16] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, "Convolutional sparse coding for image super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1823–1831.
[17] R. Hardie, K. Barnard, and E. Armstrong, "Joint map registration and high-resolution image estimation using a sequence of undersampled images," *IEEE Trans. Image Process.*, vol. 6, no. 12, pp. 1621–1633, Dec. 1997.
[18] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[20] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5197–5206.
[21] Y. Huang, W. Wang, and L. Wang, "Bidirectional recurrent convolutional networks for multi-frame super-resolution," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 235–243.
[22] Y. Huang, W. Wang, and L. Wang, "Conditional high-order boltzmann machine: A supervised learning model for relation learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4265–4273.
[23] Y. Huang, W. Wang, and L. Wang, "Instance-aware image and sentence matching with selective multimodal LSTM," *CoRR*, vol. abs/1611.05588, 2016, http://arxiv.org/abs/1611.05588
[24] M. Irani and S. Peleg, "Improving resolution by image registration," *Graph. Models Image Process.*, vol. 53, pp. 231–239, 1991.
[25] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2008, pp. 769–776.
[26] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.
[27] K. Jia, X. Wang, and X. Tang, "Image transformation based on learning dictionaries across image spaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 367–380, Feb. 2013.
[28] J. Kim, J. Kwon Lee , and K. Mu Lee , "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.
[29] 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.
[30] 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.
[31] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013, http://arxiv.org/abs/1312.4400
[32] C. Liu and D. Sun, "On Bayesian adaptive video super resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 346–360, Feb. 2014.
[33] Z. Ma, R. Liao, X. Tao, L. Xu, J. Jia, and E. Wu, "Handling motion blur in multi-frame super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5224–5232.
[34] D. Mitzel, T. Pock, T. Schoenemann, and D. Cremers, "Video super resolution using duality based TV-L 1 optical flow," in *Proc. 31st DAGM Symp. Pattern Recognit.*, 2009, pp. 432–441.
[35] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[36] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: A technical overview," *IEEE Signal Process. Mag.*, vol. 20, no. 3, pp. 21–36, May 2003.
[37] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the nonlocal-means to super-resolution reconstruction," *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 36–51, Jan. 2009.
[38] R. R. Schultz and R. L. Stevenson, "Extraction of high-resolution frames from video sequences," *IEEE Trans. Image Process.*, vol. 5, no. 6, pp. 996–1011, Jun. 1996.
[39] M. Schusterand and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[40] O. Shahar, A. Faktor, and M. Irani, "Space-time super-resolution from a single video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 3353–3360.
[41] Q. Shan, Z. Li, J. Jia, and C.-K. Tang, "Fast image/video upsampling," *ACM Trans. Graph.*, vol. 27, no. 5, 2008, Art. no. 153.
[42] 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.
[43] I. Sutskever and G. E. Hinton, "Learning multilevel distributed representations for high-dimensional sequences," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2007, pp. 548–555.
[44] H. Takeda, P. Milanfar, M. Protter, and M. Elad, "Super-resolution without explicit subpixel motion estimation," *IEEE Trans. Image Process.*, vol. 18, no. 9, pp. 1958–1975, Sep. 2009.
[45] G. Taylor, G. Hinton, and S. Roweis, "Modeling human motion using binary latent variables," in *Proc. Advances Neural Inf. Process. Syst.*, 2006, pp. 448–455.
[46] R. Timofte, V. De, and L. V. Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1920–1927.
[47] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4489–4497.
[48] R. Tsai and T. S. Huang, "Multiframe image restoration and registration," in *Proc. Advances Comput. Vis. Image Process.*, 1984, pp. 317–339.
[49] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, 1989.
[50] S. Xingjian, 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. Advances Neural Inf. Process. Syst.*, 2015, pp. 802–810.
[51] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. Advances Neural Inf. Process. Syst.*, 2014, pp. 1790–1798.
[52] 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.
[53] R. Zeyde, M. Elad, and M. Protte, "On single image scale-up using sparse-representations," in *Proc. Int. Conf. Curves Surfaces*, 2012, pp. 711–730.



Yan Huang received the BSc degree from the University of Electronic Science and Technology of China (UESTC), in 2012 and the PhD degree from the University of Chinese Academy of Sciences (UCAS), in 2017. Since July 2017, he has joined the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA) as an assistant professor. His research interests include machine learning and pattern recognition. He has published papers in the leading international conferences such as NIPS, ICCV, and CVPR.



Wei Wang received the BE degree from the Department of Automation, Wuhan University, in 2005 and the PhD degree from the School of Information Science and Engineering, Graduate University of Chinese Academy of Sciences (GUCAS), in 2011. Since July 2011, he has joined NLPR as an assistant professor. His research interests focus on computer vision, pattern recognition and machine learning, particularly on the computational modeling of visual attention, deep learning, and multimodal data analysis. He has published more than 10 papers in the leading international conferences such as CVPR and ICCV.

Liang Wang received both the BEng and MEng degrees from Anhui University, in 1997 and 2000, respectively, and the PhD degree from the Institute of Automation, Chinese Academy of Sciences (CASIA), in 2004. From 2004 to 2010, he was a research assistant at Imperial College London, United Kingdom, and Monash University, Australia, a research fellow with the University of Melbourne, Australia, and a lecturer with the University of Bath, United Kingdom, respectively. Currently, he is a full professor of the Hundred Talents Program at the National Lab of Pattern Recognition, CASIA. His major research interests include machine learning, pattern recognition, and computer vision. He has widely published in highly ranked international journals such as the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and the *IEEE Transactions on Image Processing*, and leading international conferences such as CVPR, ICCV, and ICDM. He is a senior member of the IEEE and a fellow of the IAPR.