Building an End-to-End Spatial-Temporal Convolutional Network for Video Super-Resolution

Jun Guo and **Hongyang Chao**
School of Data and Computer Science, and
SYSU-CMU Shunde International Joint Research Institute,
Sun Yat-sen University, Guangzhou, People's Republic of China

Abstract

We propose an end-to-end deep network for video super-resolution. Our network is composed of a spatial component that encodes intra-frame visual patterns, a temporal component that discovers inter-frame relations, and a reconstruction component that aggregates information to predict details. We make the spatial component deep, so that it can better leverage spatial redundancies for rebuilding high-frequency structures. We organize the temporal component in a bidirectional and multi-scale fashion, to better capture how frames change across time. The effectiveness of the proposed approach is highlighted on two datasets, where we observe substantial improvements relative to the state of the arts.

Introduction

Video Super-Resolution (VSR) is a classical problem in computer vision. This problem targets at estimating high-resolution (HR) frames from a low-resolution (LR) video sequence. To date, as the popularity of high-resolution devices such as HDTV, there is great demand for converting low-resolution sequences into high-quality high-resolution sequences, so that they can be pleasantly viewed on those devices. What's more, various visual tasks (e.g., pedestrian detection (Daniel Costea and Nedevschi 2016), action recognition (Alfaro, Mery, and Soto 2016), and video segmentation (Kundu, Vineet, and Koltun 2016)) largely rely on the quality of input videos for high performance. As a result, VSR has attracted more and more attention.

For VSR, an LR video is usually treated as a blurred, down-sampled and noisy version of the corresponding HR video. This task is highly ill-posed, as most high-frequency information has been lost during the irreversible blurring and down-sampling operations. Practical VSR approaches assume that much high-frequency information is redundant and thus may be reconstructed from LR videos. In general, these approaches can be divided into two categories: single-frame approaches, and multi-frame approaches.

Single-frame approaches mainly come from the area of image super-resolution. Given an LR video, these approaches process each video frame individually, mapping LR frames to the HR space and then concatenating them to form an HR sequence. Early works model the mapping via simple interpolation, e.g., bicubic, and Lanczos (Duchon 1979). Recent approaches are mostly learning-based. Among them, Convolutional Neural Networks (CNNs) are well received. In the pioneer work of SRCNN (Dong et al. 2014), a 3-layer CNN was constructed to learn the mapping end-to-end. VDSR (Kim, Kwon Lee, and Mu Lee 2016a) further extended SRCNN to a much deeper (20-layer) network, and has achieved the best performance among single-frame approaches. Nevertheless, single-frame approaches completely ignore valuable temporal information, whereas adjacent frames within a video usually share similar contents. If an approach can correctly capture frame variations over time, intuitively it can better reconstruct an HR frame by utilizing details of its adjacent LR frames. In consequence, single-frame approaches used to produce unsatisfactory results.

On the contrary, multi-frame approaches take the underlying relation of frames into account. Many previous works (Katsaggelos, Molina, and Mateos 2007; Babacan, Molina, and Katsaggelos 2011; Liu and Sun 2011; Ma et al. 2015) regularize the LR to HR mapping via motion-related priors. For example, Babacan et al. (2011) reconstructed HR videos subject to rotation and translation among LR frames. Liu and Sun (2011) utilized a more complex motion form, estimating optical flow, noise level, and blur kernel simultaneously from LR data. DEL (Liao et al. 2015) realized that accurately estimating motion by a single algorithm is difficult. To resolve this issue, it employed various optical flow algorithms to generate HR candidates (called "drafts") and then learned a CNN to select the best draft. DEL runs much faster, and has become the state-of-the-art multi-frame approach on VSR. Despite the success of DEL, we still find its limitations in three aspects: 1) The draft generation procedure is independent of the training of the CNN, so their powers cannot be well combined; 2) Motion is only one kind of temporal information. Other information like color variations is not well explored in DEL; 3) The CNN in DEL is not deep enough (i.e., has only 4 layers), which limits its capability of exploiting high-frequency redundancies.

To overcome these issues, in this paper, we propose a deep Spatial-Temporal CNN (STCN) as an end-to-end mapping between LR and HR video frames. Our model consists of three components: a spatial component, a temporal component, and a reconstruction component. Motivated by

VDSR, the spatial component of our STCN is a very deep CNN, aiming at discovering spatial redundancies in each input frame. This component turns inputs into highly nonlinear representations. Based on its outputs, the temporal component captures how frames change during a time period. Recurrent Neural Networks (RNNs), especially those based on LSTM (Hochreiter and Schmidhuber 1997), are widely adopted as temporal models to cope with sequential inputs. We extend LSTM to work in a bidirectional, multi-scale, and convolutional fashion, so that it is better at perceiving temporal information. Finally, the reconstruction component aggregates the outputs of the temporal component to predict an HR video. Note that our end-to-end design puts the whole model in a unified optimization framework, so we can avoid the aforementioned first issue when training. In addition, our STCN does not try to model frame relations explicitly (e.g., by optical flow). We argue that explicit modeling may be insufficient for exploring temporal information, as there are too many kinds of temporal relations (object motion, color transition, and patch similarity for instances) to be exhaustively modeled. Hence, in our STCN, temporal information is implicitly discovered via hidden layers, in order to resolve the second issue. Besides, the third issue doesn't bother our STCN either, as our STCN is organized as a very deep network. Extensive experiments prove its superior accuracy.

Related Works

Our work is mainly related to the two state of the arts, i.e., VDSR and DEL. Therefore, we focus on introducing these two approaches in the following.

Interestingly, both VDSR and DEL are based on CNNs. CNNs date back decades (LeCun et al. 1989) and have recently shown explosive successes in both high-level (Krizhevsky, Sutskever, and Hinton 2012; Sun et al. 2015) and low-level vision tasks (Dong et al. 2014). In particular, Dong et al. (2014) showed that lots of single-frame approaches for super resolution (Chang, Yeung, and Xiong 2004; Yang et al. 2010; 2012; Bevilacqua et al. 2012; Timofte, De Smet, and Van Gool 2013) could be re-interpreted as a CNN. In their work, a 3-layer network called SRCNN was built for patch extraction, non-linear LR to HR mapping and reconstruction. Kim et al. (Kim, Kwon Lee, and Mu Lee 2016a) proved that such a simple CNN had only a small receptive field and thus was not sufficient for discovering spatial context. Instead, they built a 20-layer CNN named VDSR by cascading many small filters and have achieved promising results. However, VDSR processes frames independently, and thus cannot leverage temporal information.

For VSR, multi-frame approaches are more welcomed as they can utilize temporal relations. Previous approaches (Tsai and Huang 1984; Elad and Hel-Or 2001; Farsiu et al. 2004; Katsaggelos, Molina, and Mateos 2007; Belekos, Galatsanos, and Katsaggelos 2010; Babacan, Molina, and Katsaggelos 2011; Liu and Sun 2011; Ma et al. 2015) usually involve a few components for motion estimation on LR frames. These approaches often require users to manually tune motion estimation parameters for each input. To address this problem, DEL (Liao et al. 2015) trained a CNN to automatically select the best motion estimation from a set of candidates. More precisely, DEL consists of two components. The first component proposes plentiful motion estimates, so as to generate many HR drafts, using several existing optical flow algorithms along with various parameters. After that, the second component feeds drafts into a CNN to find the best HR reconstruction. Unfortunately, as pointed out in the Introduction, these two components are independent. Hence, errors introduced by the first component are hard to be removed in the second component. Besides, DEL pays little attention to other temporal relations except for motion. Its relatively shallow CNN also imposes restrictions on its capability.

Proposed Method

Formulation

Consider an input sequence of LR frames, we first upscale it to the desired size using bicubic interpolation. Denote the interpolated LR frames as $\mathcal{X} = \{\mathbf{X}_{-T}, \cdots, \mathbf{X}_0, \cdots, \mathbf{X}_T\}$. Here T is the radius of temporal neighborhood. Our goal is to recover an HR frame $\mathbf{Y}_0$ from the corresponding LR frame $\mathbf{X}_0$, using a nonlinear mapping F. That is, we hope to find a trainable mapping which satisfies $F(\mathbf{X}_0; \mathcal{X}) \approx \mathbf{Y}_0$. Note that the output of F is not just dependent on $\mathbf{X}_0$, since some of its parameters are shared over the whole sequence $\mathcal{X}$ for exploiting temporal information. For the ease of presentation, in the following we still call $\mathbf{X}_{-T}, \cdots, \mathbf{X}_T$ as LR frames, although they have the same size as $\mathbf{Y}_0$. Especially, we refer $\mathbf{X}_0$ / $\mathbf{Y}_0$ as the target LR / HR frame.

In this paper, we develop the non-linear mapping F as a deep Spatial-Temporal CNN (STCN). Fig. 1 provides an overview for our STCN. Conceptually, our STCN has three major components, including the spatial component, the temporal component, and the reconstruction component. Next we describe them in details.

The Spatial Component

CNN-based approaches are often adopted to utilize the spatial continuity property of normal images. Inspired by VDSR, we use a series of very deep (L_S-layer) CNNs as the spatial component, to recover high-frequency details by exploiting spatial redundancies, e.g., similarity between image patches. The t-th CNN operates on the t-th frame, respectively. Each layer of a CNN extracts non-linear representations from the previous layer, and its output is further feed into the next layer to form another set of representations. The representation extraction is finished by applying convolutions on inputs, and the non-linearity is obtained by applying PReLU (He et al. 2015) on filter responses. Mathematically, the i-th layer of the t-th CNN is formulated as:

$$S_{i,t}(\mathbf{X}_t) = \text{PReLU}(\mathbf{W}_{i,t} * S_{i-1,t}(\mathbf{X}_t) + \mathbf{b}_{i,t}), \quad (1)$$

where $*$ denotes a convolution operator, $\mathbf{W}_{i,t}$ and $\mathbf{b}_{i,t}$ are the filters and biases, and $\mathbf{X}_t$ is the t-th LR frame. By stacking a large number of non-linear layers, the capability of the spatial component can be significantly strengthened, so that more complicated redundancies may be discovered.

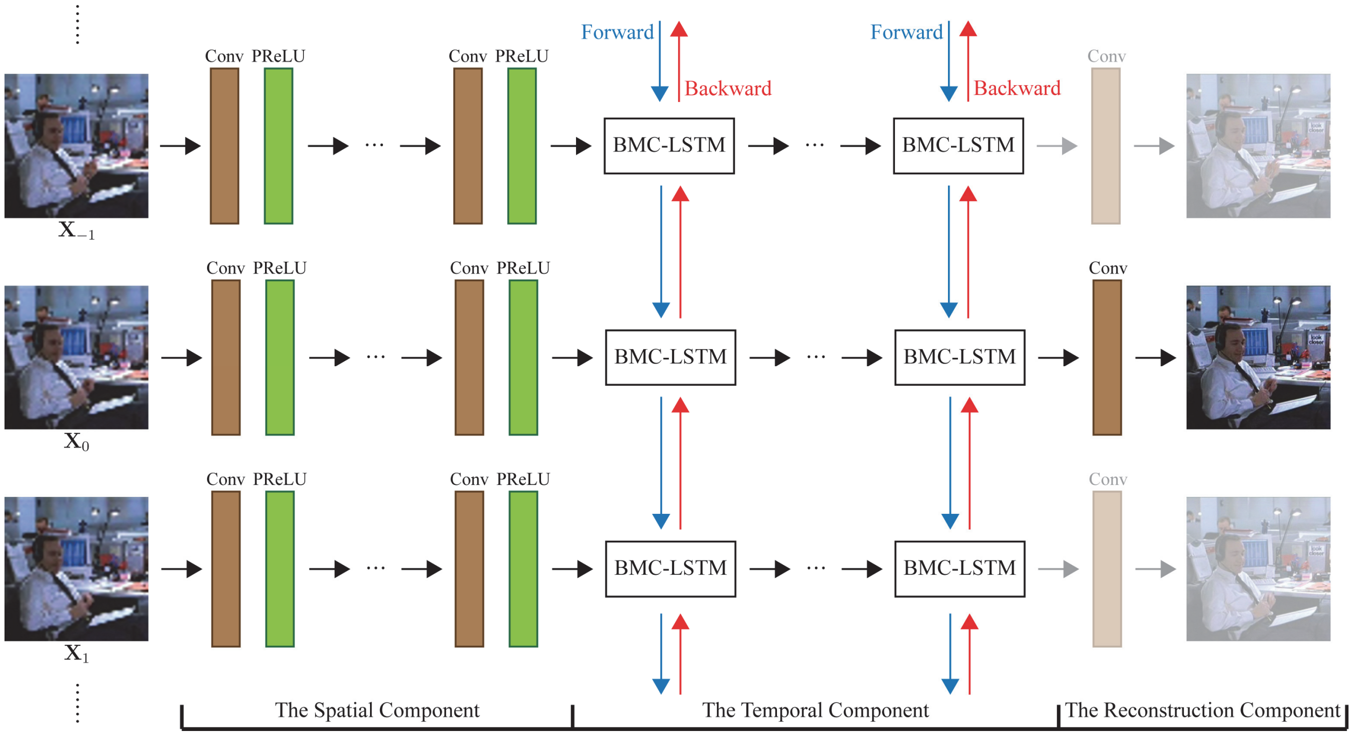

Figure 1: Our STCN contains three components. Given a series of LR frames, the spatial component captures spatial redundancies within each frame. The temporal component captures how frame changes over time. The reconstruction component maps the outputs of the temporal component back to the pixel domain to generate HR frames.

In this paper, all layers except the first are of the same type: every layer contains 64 filters of size $3 \times 3 \times 64$, i.e., each filter operates on 3×3 spatial regions across 64 channels. The first layer also contains 64 filters, but the filter size is $3 \times 3 \times C_{\text{in}}$, where C_{in} is the channel number of an input LR frame. This filter configuration is inspired by the VGGNet (Simonyan and Zisserman 2014), which showed that a deep network with very small filters was usually superior to a relatively shallow network with large filters.

The Temporal Component

The temporal component is built on top of the spatial component. It tries to implicitly discover frames variations over time with respect to the target frame. RNNs, especially those based on LSTM, have attracted significant attention in exploiting temporal information. Recently, LSTM was extended to 2D inputs (e.g., images) by introducing the convolution operator into its gate computation (Xingjian et al. 2015; Toderici et al. 2015). More precisely, this extension (literally named Convolutional LSTM, C-LSTM) replaces fully connections between weights and inputs / states inside an LSTM with convolutions. We also borrow the power of this extension. As the initial version, the temporal component is defined as a multi-layer network, with each layer containing a C-LSTM. For Layer i, let $\mathbf{Z}_{i,t}$, $\mathbf{C}_{i,t}$ and $\mathbf{H}_{i,t}$ denote its input, cell and hidden states at the t-th frame, respectively. Given the current input $\mathbf{Z}_{i,t}$, the previous cell state $\mathbf{C}_{i,t-1}$, and the previous hidden state $\mathbf{H}_{i,t-1}$, we compute the new states $\mathbf{C}_{i,t}$ and $\mathbf{H}_{i,t}$ as follows:

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{I} \\ \mathbf{O} \\ \mathbf{G} \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} (\mathbf{U}_i * \mathbf{Z}_{i,t} + \mathbf{V}_i * \mathbf{H}_{i,t-1} + \mathbf{b}_i) \quad (2)$$

$$\mathbf{C}_{i,t} = \mathbf{F} \odot \mathbf{C}_{i,t-1} + \mathbf{I} \odot \mathbf{G} \quad (3)$$

$$\mathbf{H}_{i,t} = \mathbf{O} \odot \tanh(\mathbf{C}_{i,t}) \quad (4)$$

where $\odot$ denotes element-wise multiplication, $\mathbf{U}_i$ is the filters on inputs, $\mathbf{V}_i$ is the filters on hidden states, $\mathbf{b}_i$ is the bias, and sigm denotes the sigmoid function. Note that $\mathbf{U}_i$, $\mathbf{V}_i$, and $\mathbf{b}_i$ are shared over time, so temporal relations between frames can be encoded.

Similar to the spatial component, the i-th layer of the temporal component takes the hidden state of the $(i-1)$-th layer as input: $\mathbf{Z}_{i,t} = \mathbf{H}_{i-1,t}$, except that the first layer operates on the outputs of the spatial component: $\mathbf{Z}_{1,t} = S_{L_S,t}(\mathbf{X}_t)$.

Exploiting Multi-Scale Spatial Information At first glance, it may be strange that we hope the temporal component can cope with spatial information. We argue that it is necessary to aggregate multi-scale information when frames are processed jointly. As a simple example, suppose there is an object whose size varies over time. When the spatial component processes each frame individually, it is encouraged *not* to be scale-invariant, as keeping the actual scale of the object is very important for reconstruction. Hence, when the temporal component receives outputs from the spatial

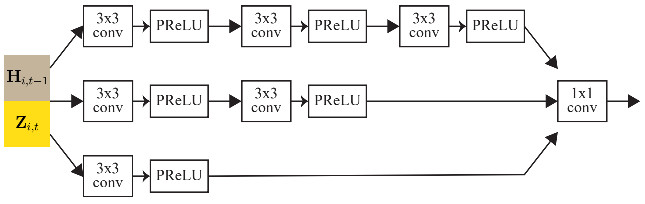

Figure 2: An overview of the transform function P in an MC-LSTM. $\mathbf{Z}_{i,t}$ and $\mathbf{H}_{i,t-1}$ are concatenated and then feed into 3×3 convolutions of various depths. The outputs are concatenated and compressed by a 1×1 convolution.

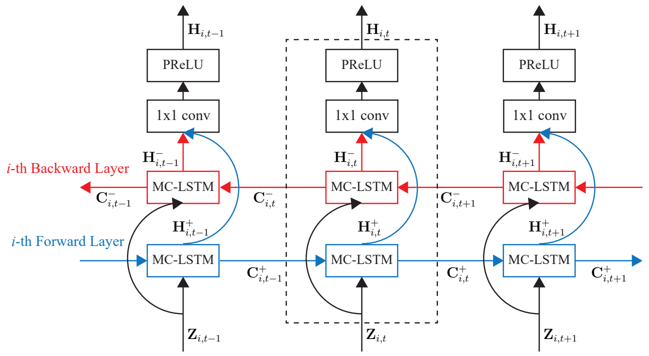

Figure 3: An overview of the BMC-LSTM. The structure surrounding by a dotted rectangle is a BMC-LSTM at the t-th frame. Its input, $\mathbf{Z}_{i,t}$, is passed to two MC-LSTMs of opposite directions, whose outputs, $\mathbf{H}^+_{i,t}$ and $\mathbf{H}^-_{i,t}$, are then aggregated to form the output of the BMC-LSTM, $\mathbf{H}_{i,t}$.

component, it needs to perform scale alignment on the object. That is, it should revise the representations of adjacent frames with respect to the target frame.

Look back on the aforementioned C-LSTM. In Eq. (2), only one type of filters are applied to inputs / hidden states. Thus, the exploration of multi-scale information is limited. To perform multi-scale analyses, we generalize Eq. (2) as:

$$\begin{pmatrix} \mathbf{F} \\ \mathbf{I} \\ \mathbf{O} \\ \mathbf{G} \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} P(\text{concat}(\mathbf{Z}_{i,t}, \mathbf{H}_{i,t-1}); \mathbf{W}_i, \mathbf{b}_i) \quad (5)$$

where "concat" is a concatenation operator along channel, and P is a transform function with weights $\mathbf{W}_i$ and biases $\mathbf{b}_i$. It can be seen that Eq. (2) is just a special case of Eq. (5).

Now the key issue is how to develop a suitable P. Here we still follow the convolutional fashion. To capture multi-scale information, a typical way is to convolve inputs (i.e., concat($\mathbf{Z}_{i,t}, \mathbf{H}_{i,t-1}$)) with various filter sizes, and then concatenate their responses. Notice that by convolving an input N times with 3×3 filters, a network can capture visual patterns of size $(2N + 1) \times (2N + 1)$. Thus, we can replace large filters with multiple small 3×3 filters to reduce computational cost. For example, a 5×5 convolution can be substituted by two 3×3 convolutions. This trick is also used in the later version of GoogLeNet (Szegedy et al. 2015). We insert PReLU after each convolution to increase non-linearity. In addition, a 1×1 convolution is placed after the concatenation of filter responses to reduce dimension, which allow the temporal component to explore multi-scale information without largely increasing the dimension of states.

We call such a Multi-scale C-LSTM as an MC-LSTM. Fig. 2 illustrates the transform function P.

Exploiting Bidirectional Temporal Information The aforementioned temporal component is unidirectional. In contrast, videos are temporally continuous in both directions: A target frame is not only similar to its previous frames, but also related to its following frames. Therefore, it is natural to integrate temporal information from the future as well as the past to reconstruct a target frame.

Bidirectional LSTM (B-LSTM) (Graves, Fernández, and Schmidhuber 2005) have been widely used in processing sequential data. To our best knowledge, this extension was only applied to traditional fully-connected LSTMs, but has not been seen on C-LSTMs. Following the idea of B-LSTM, we stack a forward MC-LSTM and a backward MC-LSTM to form a Bidirectional MC-LSTM (BMC-LSTM). More specifically, we provide the same input to two MC-LSTMs that work in opposite directions, and then concatenate their outputs (i.e., hidden states). In addition, we introduce a 1×1 convolution after concatenation to aggregate bidirectional information and keep the dimension as same as the unidirectional version. PReLU is again applied to increase non-linearity. The BMC-LSTM is depicted in Fig. 3.

In summary, after plugging the multi-scale extension and the bidirectional extension, our temporal component is reformulated as an L_T-layer network, each layer of which is composed of a BMC-LSTM. We keep the channel number of all the cell and hidden states to be 64. The sizes of filters and biases are set accordingly. We denote the output of the i-th layer at the t-th LR frame as $T_{i,t}$:

$$T_{i,t}(\mathbf{X}_t; \mathcal{X}) = \mathbf{H}_{i,t}. \quad (6)$$

We include $\mathcal{X}$ as a dependency of $T_{i,t}$ because the parameters in Eq. (5) are shared over an LR sequence.

The Reconstruction Component

As the name implies, the reconstruction component runs after the temporal component, projecting its outputs back onto the pixel domain, so that we can obtain reconstructed HR frames. We define a convolutional layer for projection[1]:

$$F(\mathbf{X}_t; \mathcal{X}) = \mathbf{W}_t * T_{L_T,t}(\mathbf{X}_t; \mathcal{X}) + \mathbf{b}_t \quad (7)$$

In this paper, the size of $\mathbf{W}_t$ is set to $3 \times 3 \times C_T$, where C_T is the channel number of $T_{L_T,t}(\mathbf{X}_t; \mathcal{X})$. The filter number is C_{in}, since the reconstructed HR frame should have the same channel number as an input LR frame.

[1]We slightly abuse the notations of weights and biases for the ease of presentation: $\mathbf{W}$ and $\mathbf{b}$ are repeatedly used in Eq. (1), Eq. (5) and Eq. (7). Nevertheless, their meanings should be clear based on context.

Training STCN

Our STCN is trained end-to-end. The objective function to be minimized is defined as:

$$\|F(\mathbf{X}_0;\mathcal{X})-\mathbf{Y}_0\|_2^2+\lambda \sum_{\mathbf{X}_i\in\mathcal{X}-\{\mathbf{X}_0\}} \|F(\mathbf{X}_i;\mathcal{X})-\mathbf{Y}_i\|_2^2 \quad (8)$$

Different from DEL, we not only require our STCN to correctly reconstruct the target frame, but also encourage it to be accurate when predicting the other HR frames. In this way, we can make full use of training data, in contrast to DEL which completely neglects non-target HR frames during training. Initially, we set λ to 1 for utilizing all training samples. As training progresses, λ decays to 0 to boost the performance on the target frame.

In practice, we find that replacing $\mathbf{Y}_0$ and $\mathbf{Y}_i$ in Eq. (8) with $(\mathbf{Y}_0-\mathbf{X}_0)$ and $(\mathbf{Y}_i-\mathbf{X}_i)$ respectively makes the training procedure converge faster, and produces slightly better results. An intuitive explanation is, when we train STCN to directly predict HR frames, STCN needs to remember most pixel values of input LR frames through the whole network, since an LR frame and its corresponding HR frame are expected to be similar. Such long-term memory can easily lead to gradient vanishing or explosion (Hochreiter and Schmidhuber 1997). Instead, when STCN is trained towards the residuals between LR and HR frames, long-term memory is no longer required, leading to simpler optimization.

Ideally, all three components of our STCN should be trained jointly. However, it would result in huge computational cost, especially when the radius of temporal neighborhood T is large. Hence, we employ the following training strategy: only the CNN of the spatial component that operates on the target frame is always updated together with the temporal component and the reconstruction component. For the remaining CNNs of the spatial component, in each iteration we randomly pick up one and update its parameters, while at the same time the others are unchanged.

We adopt Adam (Kingma and Ba 2014) to train our STCN. We begin with a step size 10^{-4} and then decrease it by a factor of 10 when the validation error stops improving. The step size of Adam is reduced twice prior to termination. The other hyper-parameters of Adam are set according to the guideline provided by the Adam paper. We train a specific network with batch size 64 for each upscaling factor.

Experiments

In this section, we present experimental results to demonstrate the effectiveness of the proposed STCN for VSR.

For fair comparisons with existing works, we follow the protocol of DEL to conduct the first experiment. Three benchmark sequences, i.e., *penguin*, *temple* and *city*, are used as our first test set (we denote it as "TriVideos" in this paper). The training set is built by collecting 160 video sequences from 26 high-quality 1080p HD video clips. All videos in the training / test set are trimmed to contain 31 consecutive frames where the central one is regarded as the target frame. To generate LR frames, HR frames are first blurred by Gaussian filters for anti-aliasing and then down-sampled. For more details please refer to the DEL paper.

Table 1: Comparison on the TriVideos Dataset

Approach	Metric	*penguin*	*temple*	*city*
BayesSR	PSNR	29.53	29.01	25.49
	SSIM	0.9451	0.9375	0.7979
DEL	PSNR	31.87	30.23	24.89
	SSIM	0.9483	0.9504	0.7610
DRCN	PSNR	32.50	30.62	25.57
	SSIM	0.9510	0.9567	0.8025
VDSR	PSNR	32.59	30.64	25.56
	SSIM	0.9512	0.9568	0.8022
STCN	PSNR	**33.64**	**31.16**	**26.01**
	SSIM	**0.9541**	**0.9627**	**0.8141**

We note that TriVideos may not be a conclusive test set due to the limited number of test samples. To further increase credibility, we also conduct experiments on the larger Hollywood2 Scenes dataset (Marszalek, Laptev, and Schmid 2009). This is one of the most famous datasets in the video community, having 570 training sequences and 582 test sequences. Similarly, for each video we only take the middle 31 consecutive frames for training / testing, with the central frame being the target frame. Blurring and down-sampling are used to generate LR frames. Following VDSR, on Hollywood2 we only consider the luminance channel (in YCbCr color space). The two chrominance channels are bicubic interpolated only for display, but not for training / testing.

For all experiments, the validation set is built by re-using frames that are trimmed from the training set.

Implementation Details

Network Input In the training phase, we uniformly sample (41×41)-pixel patches from (interpolated) LR and HR frames with stride 28. Smaller strides have been tried but no significant performance gains were observed. In the testing phase, STCN runs as a fully convolutional network (Long, Shelhamer, and Darrell 2015) for full-frame predictions.

Network Structure The layer numbers of the spatial and temporal components, i.e., L_S and L_T, are set to 20 and 3, respectively. If not specified, the temporal neighborhood radius T is set to 2, i.e., only 5 adjacent frames are used as inputs, though more frames can be utilized theoretically. Zero-padding is applied before convolutions to keep the output size equal to the input size. All filters are initialized using He et al.'s initializer (2015). All states are initialized to 0.

Quantitative Evaluation on TriVideos

We mainly compare results with the state-of-the-art single-frame approach, VDSR, and the state-of-the-art multi-frame approach, DEL, under the upscaling factor 4. Two recent approaches, the single-frame-based DRCN (Kim, Kwon Lee, and Mu Lee 2016b) and the multi-frame-based BayesSR (Liu and Sun 2011), are also included. There are some other recent works like EPSCN (Shi et al. 2016) for VSR. But these works usually focuses on speed, and their reported results are worse than VDSR on several datasets. Therefore, we don't include them here.

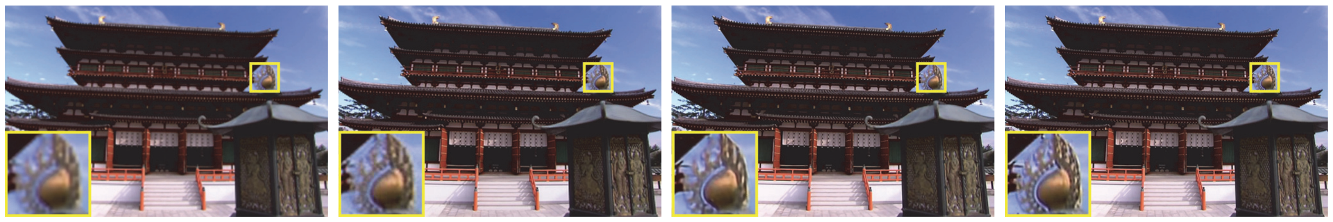
(a) Bicubic (b) DEL (c) STCN (d) Groundtruth

Figure 4: Comparison on Sequence *temple* from TriVideos under the upscaling factor 4.

Table 2: Comparison on the Hollywood2 Dataset

Approach	Metric	x2	x3	x4	x5
Bicubic	PSNR	37.25	33.73	31.66	30.31
	SSIM	0.9641	0.9248	0.8834	0.8451
DEL	PSNR	40.53	36.41	34.01	31.89
	SSIM	0.9744	0.9499	0.9176	0.8793
VDSR	PSNR	41.06	36.61	34.20	32.27
	SSIM	0.9785	0.9514	0.9204	0.8853
STCN	PSNR	**41.51**	**37.03**	**34.58**	**32.56**
	SSIM	**0.9804**	**0.9546**	**0.9259**	**0.8902**
STCN(SS)	PSNR	41.27	36.78	34.35	32.38
	SSIM	0.9793	0.9524	0.9229	0.8874
STCN(Uni)	PSNR	41.39	36.89	34.46	32.43
	SSIM	0.9797	0.9538	0.9241	0.8886

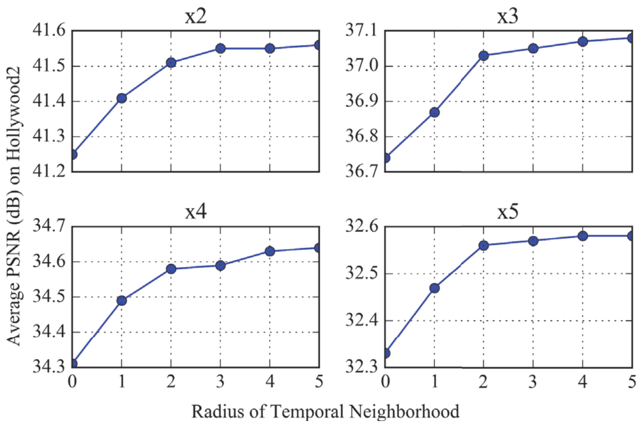

Figure 5: Average PSNR (dB) on the test set of Hollywood2 for various temporal neighborhood radii.

As shown in Table 1, our results are promising under both two objective fidelity metrics. Specifically, the PSNR gains achieved by STCN are 1.05 dB, 0.52 dB, and 0.44 dB, respectively, in comparison to the next best approach, on three test videos. DRCN performs almost the same as VDSR. This is reasonable, as their network structures are nearly equivalent. BayesSR is inferior to the other approaches in general. Interestingly, DEL seems to be worse than VDSR, though DEL can leverage temporal information. This phenomenon demonstrates the insufficiency of the overly simple CNN in DEL. We present the super-resolution results for Sequence *temple* in Fig. 4. It can be seen that our STCN generates more details and sharper edges than the other approaches, without introducing any obvious artifact.

Quantitative Evaluation on Hollywood2

Here we conduct more experiments on Hollywood2, under 4 upscaling factors (2, 3, 4 and 5). Since BayesSR doesn't provide codes for evaluation, and its performance is inferior to the other approaches on TriVideos, in this section we don't report its results. DRCN is also skipped as it performs on par with VDSR on TriVideos.

The experimental results are shown in Rows 2∼5 of Table 2. From the perspective of PSNR, our STCN consistently outperforms the other approaches by a large margin (≈ 0.4 dB on average). A similar trend can be observed for SSIM, as well. These quantitative results verify the effectiveness of the proposed approach. We show some qualitative results from different approaches under the upscaling factor 5 in Fig. 6. As can be observed, STCN recovers more structures. The reconstructed contours are cleaner and more vivid in STCN, again demonstrating that our approach is not only superior in objective fidelity metric, but is also visually appealing. Note that, only our approach perfectly recovers the two horizontal lines at the top of the pole in Sequence 153. This is pretty interesting, as both lines are almost indistinguishable in the bicubic interpolation result (i.e., Row 3 and Col 1 of Fig. 6), which is also the target LR frame. It indicates that our STCN can successfully leverage information from adjacent frames to help reconstruction.

Temporal Neighborhood Radius To investigate the influence of the temporal neighborhood radius, we evaluate STCN under various Ts, and display the results in Fig. 5. As can be seen, performance grows as radius increases in general. But the performance gain seems to become marginal when $T >= 3$. This reflects the difficulty in exploring long-term temporal information, and is reserved for future study. We regard $T = 2$ as a good balance between reconstruction quality and computational cost.

Multi Scales We develop a variant of STCN (denoted as "STCN(SS)") whose temporal component only works in single scale. That is, we replace the two MC-LSTMs in a BMC-LSTM by two C-LSTMs, and show the results in Row 6 of Table 2. Compared with Row 5 of Table 2, it is clear that

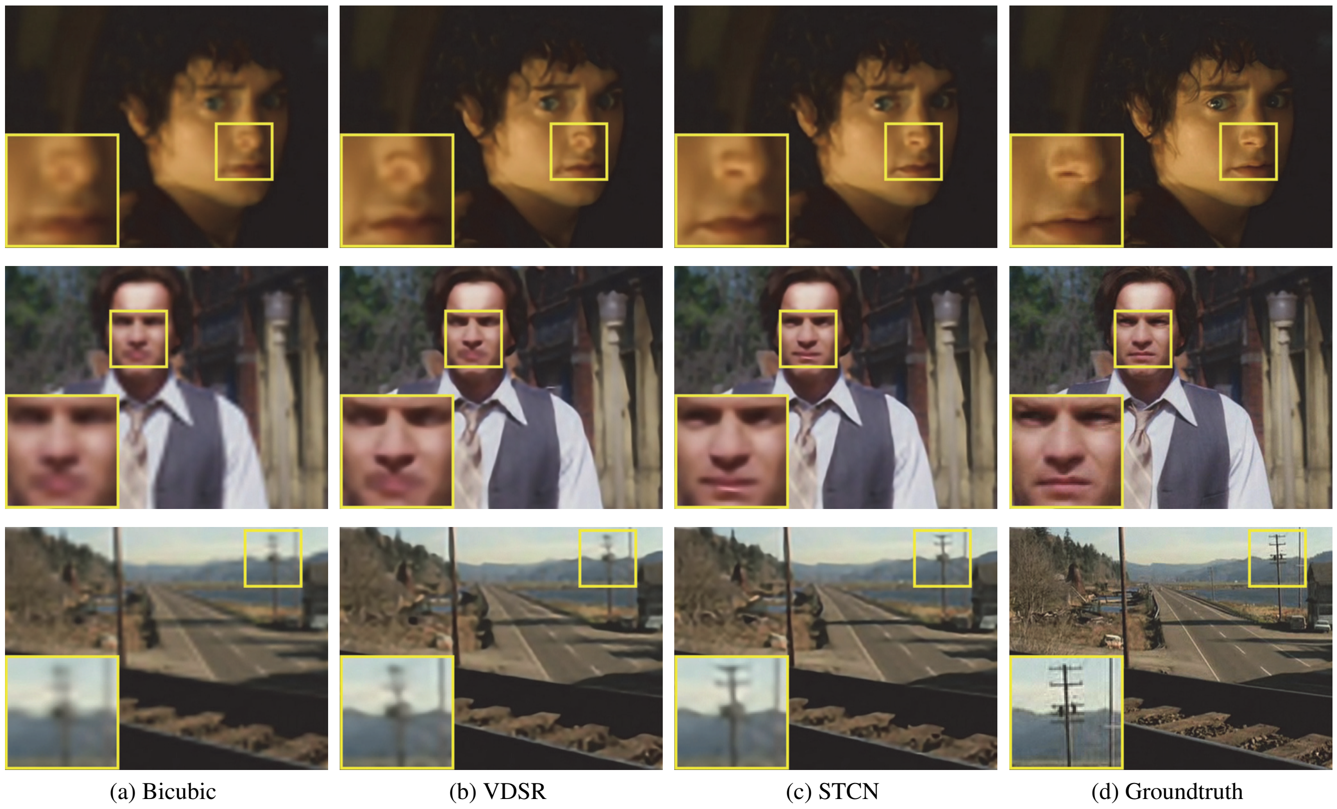

Figure 6: Comparison on Sequence 005 (Row 1), 049 (Row 2) and 153 (Row 3) from Hollywood2 under the upscaling factor 5.

using only one scale leads to performance drop. This experiment confirms the importance of our multi-scale design.

Bidirection In order to verify the effectiveness of bidirection, we introduce another variant of STCN which only employs unidirectional MC-LSTMs in its temporal component. For a fair comparison, for each target frame, this variant (denoted as “STCN(Uni)”) takes the target frame and the 4 consecutive frames previous to it as input. In this way, both STCN and STCN(Uni) use 5 LR frames for reconstruction. The last row of Table 2 shows the performance of STCN(Uni). In comparison to Row 5 of Table 2, we can conclude that integrating information from the future is essential for good reconstruction. This is quite intuitive. Given a target frame, the frames that are most related to it are those closest to it, in general. Hence, information within 2 previous frames and 2 next frames are usually more valuable than that within 4 previous frames.

Conclusions

In this paper, we systematically studied how to build an effective network for video super-resolution. The proposed approach, STCN, learns an end-to-end mapping between low- and high-resolution frames. We presented a very deep CNN to fully utilize spatial information. We also designed a bidirectional and multi-scale convolutional LSTM to implicitly leverage temporal information. Experimental results demonstrated the promise of STCN. In future, we would like to examine our STCN on other video restoration tasks like denoising and compression artifacts reduction.

Acknowledgments

This work is partially supported by NSF of China, under Grant 61672548, 61173081, and the Guangzhou Science and Technology Program, China, under Grant 201510010165.

References

Alfaro, A.; Mery, D.; and Soto, A. 2016. Action recognition in video using sparse coding and relative features. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

Babacan, S. D.; Molina, R.; and Katsaggelos, A. K. 2011. Variational bayesian super resolution. *IEEE Transactions on Image Processing* 20(4):984–999.

Belekos, S. P.; Galatsanos, N. P.; and Katsaggelos, A. K. 2010. Maximum a posteriori video super-resolution using a new multichannel image prior. *IEEE Transactions on Image Processing* 19(6):1451–1464.

Bevilacqua, M.; Roumy, A.; Guillemot, C.; and Alberi-Morel, M. L. 2012. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proceedings of the British Machine Vision Conference*. BMVA Press.

Chang, H.; Yeung, D.-Y.; and Xiong, Y. 2004. Super-resolution through neighbor embedding. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, I–I. IEEE.

Daniel Costea, A., and Nedevschi, S. 2016. Semantic channels for fast pedestrian detection. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

Dong, C.; Loy, C. C.; He, K.; and Tang, X. 2014. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision*, 184–199. Springer.

Duchon, C. E. 1979. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology* 18(8):1016–1022.

Elad, M., and Hel-Or, Y. 2001. A fast super-resolution reconstruction algorithm for pure translational motion and common space-invariant blur. *IEEE Transactions on Image Processing* 10(8):1187–1193.

Farsiu, S.; Robinson, M. D.; Elad, M.; and Milanfar, P. 2004. Fast and robust multiframe super resolution. *IEEE transactions on image processing* 13(10):1327–1344.

Graves, A.; Fernández, S.; and Schmidhuber, J. 2005. Bidirectional lstm networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, 799–804. Springer.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 1026–1034.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Katsaggelos, A. K.; Molina, R.; and Mateos, J. 2007. Super resolution of images and video. *Synthesis Lectures on Image, Video, and Multimedia Processing* 1(1):1–134.

Kim, J.; Kwon Lee, J.; and Mu Lee, K. 2016a. Accurate image super-resolution using very deep convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

Kim, J.; Kwon Lee, J.; and Mu Lee, K. 2016b. Deeply-recursive convolutional network for image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Kundu, A.; Vineet, V.; and Koltun, V. 2016. Feature space optimization for semantic video segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1(4):541–551.

Liao, R.; Tao, X.; Li, R.; Ma, Z.; and Jia, J. 2015. Video super-resolution via deep draft-ensemble learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 531–539.

Liu, C., and Sun, D. 2011. A bayesian approach to adaptive video super resolution. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 209–216. IEEE.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440.

Ma, Z.; Liao, R.; Tao, X.; Xu, L.; Jia, J.; and Wu, E. 2015. Handling motion blur in multi-frame super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5224–5232.

Marszalek, M.; Laptev, I.; and Schmid, C. 2009. Actions in context. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 2929–2936. IEEE.

Shi, W.; Caballero, J.; Huszar, F.; Totz, J.; Aitken, A. P.; Bishop, R.; Rueckert, D.; and Wang, Z. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sun, L.; Jia, K.; Yeung, D.-Y.; and Shi, B. E. 2015. Human action recognition using factorized spatio-temporal convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 4597–4605.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*.

Timofte, R.; De Smet, V.; and Van Gool, L. 2013. Anchored neighborhood regression for fast example-based super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision*, 1920–1927.

Toderici, G.; O'Malley, S. M.; Hwang, S. J.; Vincent, D.; Minnen, D.; Baluja, S.; Covell, M.; and Sukthankar, R. 2015. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085*.

Tsai, R., and Huang, T. S. 1984. Multiframe image restoration and registration. *Advances in computer vision and Image Processing* 1(2):317–339.

Xingjian, S.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-k.; and Woo, W.-c. 2015. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems*, 802–810.

Yang, J.; Wright, J.; Huang, T. S.; and Ma, Y. 2010. Image super-resolution via sparse representation. *IEEE transactions on image processing* 19(11):2861–2873.

Yang, J.; Wang, Z.; Lin, Z.; Cohen, S.; and Huang, T. 2012. Coupled dictionary training for image super-resolution. *IEEE Transactions on Image Processing* 21(8):3467–3478.