Omniscient Video Super-Resolution

Peng Yi[1], Zhongyuan Wang*[1], Kui Jiang[1], Junjun Jiang[2,3], Tao Lu[4], Xin Tian[5], and Jiayi Ma[5]

[1]National Engineering Research Center for Multimedia Software, School of Computer Science, Wuhan University [2]School of Computer Science and Technology, Harbin Institute of Technology [3]Peng Cheng Laboratory [4]School of Computer Science and Engineering, Wuhan Institute of Technology
[5]Electronic Information School, Wuhan University
{yipeng, kuijiang, xin.tian}@whu.edu.cn, {wzy_hope, junjun0595}@163.com,
{lutxyl, jyma2010}@gmail.com

Abstract

Most recent video super-resolution (SR) methods either adopt an iterative manner to deal with low-resolution (LR) frames from a temporally sliding window, or leverage the previously estimated SR output to help reconstruct the current frame recurrently. A few studies try to combine these two structures to form a hybrid framework but have failed to give full play to it. In this paper, we propose an omniscient framework to not only utilize the preceding SR output, but also leverage the SR outputs from the present and future. The omniscient framework is more generic because the iterative, recurrent and hybrid frameworks can be regarded as its special cases. The proposed omniscient framework enables a generator to behave better than its counterparts under other frameworks. Abundant experiments on public datasets show that our method is superior to the state-of-the-art methods in objective metrics, subjective visual effects and complexity.

1. Introduction

Super-Resolution (SR) aims at reconstructing high-resolution (HR) images from the corresponding low-resolution (LR) images. As the most basic problem in SR, single image super-resolution (SISR) has been relatively studied thoroughly, where under a unified framework, the researchers only have to design different kinds of convolutional neural networks (CNNs) [4, 14, 32, 31, 3] to solve this issue. Based on SISR, video super-resolution (VSR) has also been developed, albeit a lot of works [1, 22, 19, 27, 29, 11, 10] have been proposed, *there is not a unified framework being dominant in VSR yet*. Figure 1 illustrates dozens of state-of-the-art (SOTA) VSR methods in terms of performance and speed.

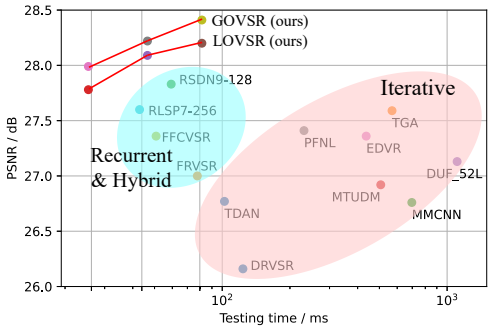
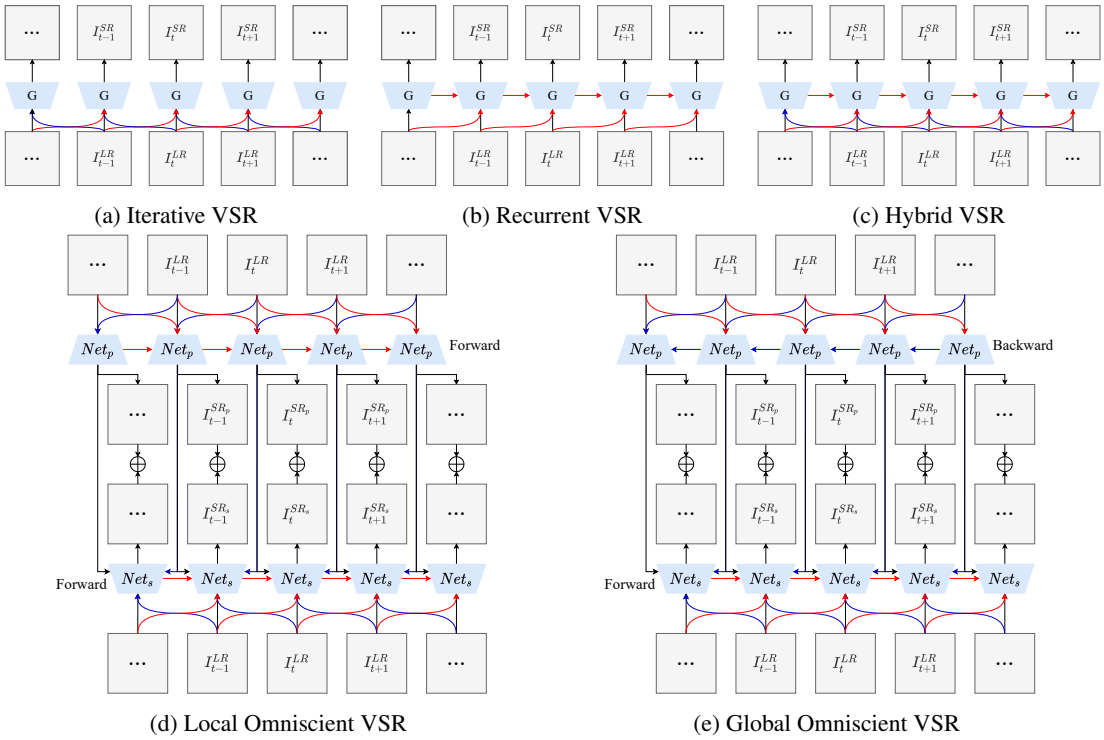


Figure 1: Performances on Vid4 [1] dataset and time costs. The Red circle denotes iterative methods, and the blue circle indicates recurrent and hybrid methods. Please refer to Table 4 and Table 6 for more details.

As SISR requires only one input image, most SISR methods focus on exploring different generator networks to extract features from this one image under a unified framework. Nevertheless, since VSR involves consecutive video frames as input, different schemes for handling the temporal information have emerged. We demonstrate different kinds of frameworks for VSR in Figure 2.

As illustrated in Figure 2(a), most recent VSR methods [1, 22, 25, 30, 12, 24, 29, 23, 11] apply an iterative manner to deal with LR frames from a temporally sliding window, where we only show the case of window size as 3. Given a sequence of video frames, the iterative framework considers the whole VSR processing as multiple independent sub-processes. Theoretically, these sub-processes are not temporally correlated and can be handled simultaneously, which means they enjoy the advantage of parallel comput-

*Corresponding author. Code: https://github.com/psychopa4/OVSR.



(a) Iterative VSR (b) Recurrent VSR (c) Hybrid VSR

(d) Local Omniscient VSR (e) Global Omniscient VSR

Figure 2: Different kinds of frameworks for VSR, where "G" represents the generator network. Red, black and blue arrows denote information from the past, present and future respectively.

ing [29]. However, the iterative framework can only obtain more neighboring LR frames by increasing the window size but omits the previously estimated SR output, which is the exact reason that prevents it from a better performance.

As demonstrated in Figure 2(b), the recurrent framework [19, 10] processes video frames by the order, while it can never exploit the subsequent frames to assist with recovering the current frame, which has limited its potential. Although a few studies [5, 27] have tried to combine these two frameworks to form a hybrid framework, as shown in Figure 2(c), it can only receive the estimated hidden states from the past, and they have not achieved satisfying results.

The recurrent and hybrid frameworks only leverage previous hidden states, which inspires us to wonder *what if we further try to involve the hidden states from the present and future*. To this end, we propose the omniscient framework. Specifically, we integrate two sub-networks: a precursor network Net_p and a successor network Net_s into the omniscient framework. The successor network inherits the hidden states generated by the precursor network, and thus manages to leverage the *LR frames and hidden states from the past, present and future*. As shown in Figure 2(d) and Figure 2(e), the omniscient framework can be further divided into two categories: local omniscient and global omniscient. Local omniscient framework processes video frames unidirectionally while global omniscient framework does it bidirectionally. The global omniscient framework enables any LR frame to receive information from all other frames in a same video sequence, however, it is not appropriate for delay-sensitive real-time tasks like live broadcasts, where the local omniscient framework suits well instead.

Overall, in this paper, we propose a more generic omniscient framework to exploit both LR frames and estimated hidden states from the past, present and future. In fact, as shown in Figure 2, *the iterative, recurrent and hybrid frameworks can be regarded as the special counterparts of our proposed omniscient framework.* We have explored a same kind of generator network under different frameworks, and we have found the omniscient framework (local and global) superior to the existing iterative, recurrent and hybrid frameworks. Our models surpass other SOTA methods in both performance and complexity, and thus *we hope this framework will become a standard framework in VSR, under which researchers are free to design more effective generator networks, explicit or implicit motion information capturing modules, or loss functions to tap its potential.*

2. Related Work

2.1. Iterative Video Super-Resolution

With the development of deep learning, a lot of CNN-based SISR methods, *e.g.* SRCNN [4], VDSR [14], have

emerged, from which some early VSR methods [1, 13, 17] are inspired and simply apply the architectures from SISR to VSR. These methods consider VSR as a multi-input counterpart of SISR and employ the iterative framework to solve it. These iterative methods can be further divided into two categories according to their generator networks: iterative [18, 12, 24, 29, 23, 11, 28] and recurrent [22, 25, 30, 7]. Particularly, RVSR-LTD [18] designs a temporal adaptive network with a pyramid structure, and DUFVSR [12] proposes a 3D CNN with dynamic upsampling filters. EDVR [24] has designed a pyramid, cascading and deformable (PCD) alignment module with a temporal and spatial attention (T-SA) fusion module to achieve wonderful results, however, its training resource requirement is prohibitive. PFNL [29] proposes a progressive fusion network with a non-local correlations extraction module, which has obtained an amazing result in terms of both performance and complexity. TDAN [23] devises a temporally-deformable alignment network, and TGA [11] utilizes a hybrid module with both 2D and 3D residual blocks for inter-group fusion.

Another group of iterative-based methods adopts recurrent generator networks to transfer temporal correlations instead. DRVSR [22] utilizes ConvLSTM [21] module for capturing long-range temporal information, based on which MMCNN [25] designs a multi-memory residual block to enhance the memory ability. MTUDM [30] proposes an ultra-dense memory residual block to build a shallower but wider network. RBPN [7] extends [6] to video SR, which sends LR frames into a projection module step by step. *In all, these iterative methods have been concentrating on designing more and more complicated generator networks but with huge computational costs in the meantime.*

2.2. Recurrent and Hybrid Video Super-Resolution

As to the recurrent methods, FRVSR [19] proposes a frame-recurrent network to leverage the last rebuilt SR frame for reconstructing the current frame, which is fast but not robust enough. RSDN [10] designs a dual-channel network to learn the structures and details of frames. *Still, these recurrent methods refuse the assist from subsequent LR frames.*

FFCVSR [27] and RLSP [5] have tried to form a hybrid framework but could not achieve a wonderful result because 1) the hybrid framework still cannot exploit the estimated hidden states from the present and future, and 2) their naive network designs have turned them against maximizing their potential.

3. Method

3.1. Omniscient Video Super-Resolution

As illustrated in Figure 2, the hybrid VSR combines the idea of iterative VSR and recurrent VSR, which adopts both neighboring LR frames and a previously estimated SR output as source information. Theoretically, the neighboring LR frames provide *the most basic spatial-temporal information in LR space*, and the estimated SR output can reserve more *temporally correlated information connected to the HR space*. Thus, it is natural to combine them to fully exploit the spatial-temporal correlations.

Based on the above analysis, we wonder what if we further introduce the estimated SR outputs from the present and future. However, after long deliberation, we reckon that idea is not realizable if only processing through the video frames in a single time like the iterative, recurrent, or hybrid VSR. Eventually, as shown in Figure 2(d) and Figure 2(e), we manage to design two sub-networks: a precursor network Net_p and a successor network Net_s to form the OVSR framework. The precursor network first flows through the LR frames to generate the SR frames and hidden states of all time steps, and then the successor network reconstructs all SR frames with the assist of the corresponding LR frames and estimated hidden states. We further add the SR frames generated both by the precursor and successor for refinement to reconstruct the final SR output.

According to the direction of the precursor network and successor network, the OVSR can be further categorized as local omniscient VSR (LOVSR) and global omniscient VSR (GOVSR). Both Net_p and Net_s of LOVSR process video frames in the same direction, which means it can only utilize the past and present information, as well as a limited number of frames (2 in Figure 2(d)) in the future, which is why we call it "local omniscient". Then, we design the GOVSR by reversing the direction of Net_p subtly, through which any LR frame gets access to all other frames in a same video sequence, and thus it is called "global omniscient". Although the GOVSR is capable of leveraging all frames to assist in rebuilding one frame at any time step, it relies heavily on future information. Nevertheless, frames too far in the future are actually not accessible in some delay-sensitive real-time tasks like live broadcasts and online meetings, where the LOVSR suits well. In all, *LOVSR suits online VSR and GOVSR is more appropriate for offline VSR.* Still, the omniscient VSR, no matter LOVSR or GOVSR, can exploit both the LR frames and estimated hidden states from the past, present and future. Recently, BasicVSR [2] has also proposed a bidirectional framework that suits offline VSR, while ignoring the online VSR situation.

It is worth mentioning that the proposed omniscient framework *does not require a specific structure of the generator network, instead, any kind of existing generator networks can be easily inserted into the architecture illustrated in Figure 3, as long as they satisfy the form shown in Equation* (1) *and Equation* (2).

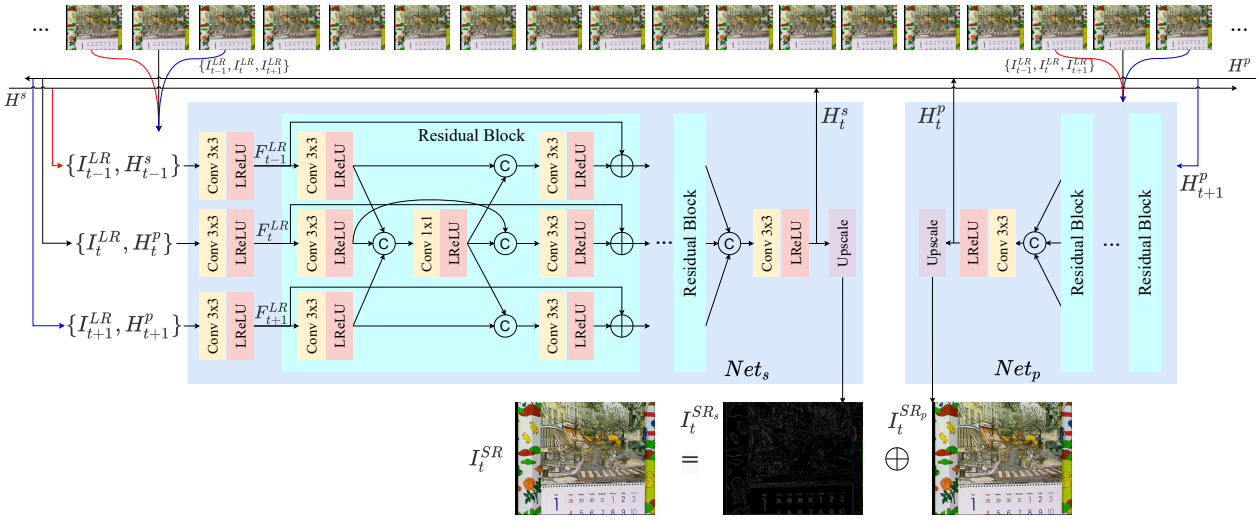


Figure 3: The whole architecture of our model under the global omniscient framework, and the local omniscient case can be inferred according to Figure 2(d). The "$\bigoplus$" represents element-wise add and "ⓒ" denotes concatenation along the channel axis. Red, black and blue arrows denote information from the past, present and future respectively.

3.2. Network Design

We then elaborate on the specific network design of our model under the omniscient framework. Although our proposed OVSR manages to leverage more source information theoretically, it still requires a well-designed structure of the generator network to fulfill its potential practically. Fortunately, we learn a progressive fusion residual block (PFRB) from PFNL [29], which is sophisticated and shows to be both effective and efficient.

To combine the PFRB with the omniscient framework, we ameliorate it to embody 3 channels. As illustrated in Figure 3, we show the architecture of our model under GOVSR, and the LOVSR case can be inferred. Because the precursor and successor networks share a similar structure, we only need to introduce the successor network. We adjust the PFRB to embody 3 channels, which contain information from the past, present and future respectively. We first adopt one convolutional layer to fuse the corresponding LR frame I_t^{LR} and hidden state H_t to obtain a feature F_t^{LR}. Correspondingly, these 3 features contain information from the past, present and future respectively. Then, in the residual blocks, features from these 3 channels are extracted both independently and merged together, in which intra-frame spatial correlations and inter-frame temporal correlations are fully exploited [29]. At the end of the network, features from these 3 channels are concatenated and processed by a 3×3 convolutional layer to obtain the updated hidden state H_t^s. The H_t^s is upscaled to $I_t^{SR_s}$, and then added by the SR frame from the precursor $I_t^{SR_p}$ to reconstruct the final SR output I_t^{SR}, where the upscale module is composed of 2 convolutional layers each followed by a sub-pixel convolution layer [20]. We set a Leaky ReLU activation [9] after every convolutional layer (except the last one in the upscale module), with parameter $\alpha = 0.2$.

The precursor network can be described as

$$I_t^{SR_p}, H_t^p = Net_p(\{I_{t-1}^{LR}, I_t^{LR}, I_{t+1}^{LR}\}, H_{t+1}^p), \quad (1)$$

where $\{I_{t-1}^{LR}, I_t^{LR}, I_{t+1}^{LR}\}$ stand for the neighboring LR frames, $I_t^{SR_p}$ and H_t^p denote the SR frame and hidden state generated by the precursor. Note that H_{t+1}^p is for GOVSR, which should be H_{t-1}^p for LOVSR instead. The successor network can be described as

$$I_t^{SR_s}, H_t^s = Net_s(\{I_{t-1}^{LR}, I_t^{LR}, I_{t+1}^{LR}\}, \{H_{t-1}^s, H_t^p, H_{t+1}^p\}), \quad (2)$$

where $I_t^{SR_s}$ and H_t^s denote the SR frame and hidden state generated by the successor. We add the SR frames generated by Net_p and Net_s to form the final output.

$$I_t^{SR} = I_t^{SR_s} + I_t^{SR_p}, \quad (3)$$

because Net_s inherits Net_p, we naturally restrict Net_p to mainly learn the low-frequency structures and Net_s to study the high-frequency details within frames. We adopt the Charbonnier loss function [16] to form the loss function:

$$\mathcal{L} = \sqrt{(I_t^{HR} - I_t^{SR})^2 + \varepsilon^2} + \alpha\sqrt{(I_t^{HR} - I_t^{SR_p})^2 + \varepsilon^2}, \quad (4)$$

where I_t^{HR} denotes the original HR frame, α is set to adjust the weight of the precursor network, and ε is empirically set to 10^{-3}.

In all, our model fully exploits the spatio-temporal correlations contained in the LR frames and estimated hidden states from the past, present and future.

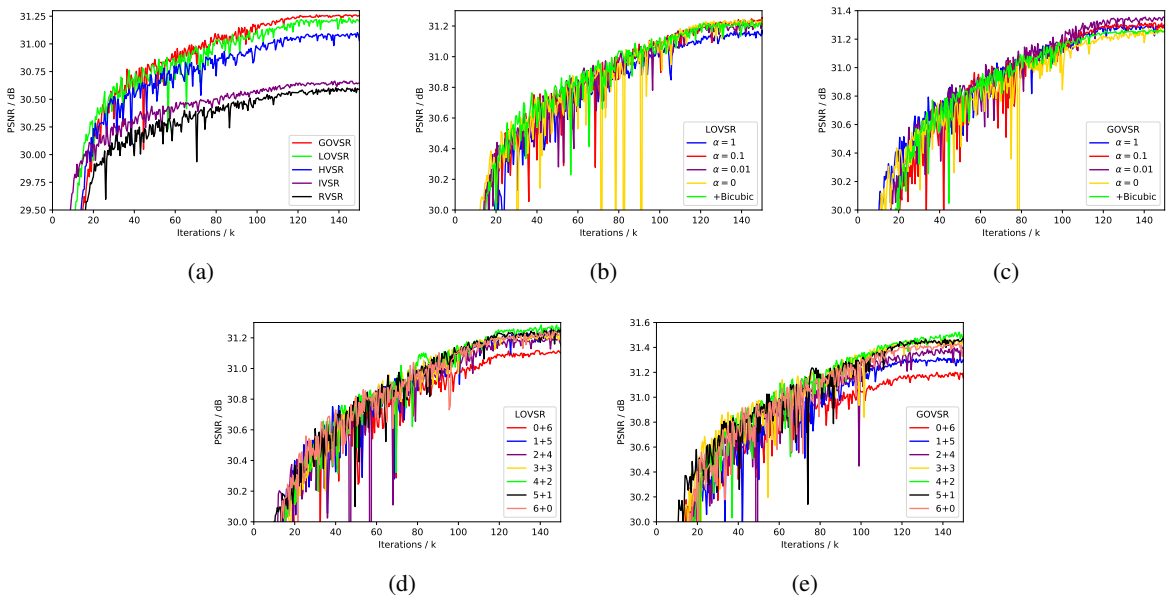


Figure 4: (a) Similar generator networks trained under different VSR frameworks. (b)(c) Adjust the weight of $I_t^{SR_p}$ during training by changing α in Equation (4). (d)(e) Adjust the proportion of Net_p and Net_s, where "4+2" means setting 4 residual blocks in Net_p and 2 in Net_s.

4. Experiments

4.1. Implementation Details

We first adopt a public MM522 dataset [25] for training, which contains 522 32-frame sequences with various scenes. During training, we employ 20 video sequences from [29] for evaluation. When trained on the MM522 dataset, we test the models on Vid4 [1] and UDM10 [30] testing datasets. We also conduct experiments on another public training dataset Vimeo-90K [26], and test the models on its testing dataset Vimeo-90K-T (please see the supplementary material). Like in [12, 24, 29, 10], we adopt Gaussian blur with $\sigma = 1.6$ and then $4\times$ down-sampling scheme.

We use Adam [15] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the batch size as 16 and input LR size as 64×64. We only feed 7 consecutive frames for the ablation study, and to train our final models as well as rebuild other SOTA methods, we set 2 additional frames for feeding the I_{t-1}^{LR}, H_{t-1} in the beginning and I_{t+1}^{LR}, H_{t+1} at the end [5]. The initial learning rate is 1×10^{-3} and decays linearly to 1×10^{-4} after 120 K iterations, which keeps the same until 200 K iterations. Then, the learning rate is further decayed to 5×10^{-5} and 1×10^{-5} until converged. The whole training process lasts about 300 K - 400 K iterations. We conduct experiments on Python 3.6, PyTorch 1.6 and NVIDIA RTX 2080Ti GPUs.

Table 1: Performances, parameters, FLOPs and testing time costs of different models. The PSNR is evaluated on 20 sequences from [29], while FLOPs and testing time costs are evaluated with respect to 1280×720 HR frame under $4\times$ SR.

Model	IVSR	RVSR	HVSR	LOVSR	GOVSR
Parameter (M)	1.864	1.866	1.868	1.897	1.897
FLOPs (G)	107.780	107.696	107.796	109.746	109.746
Testing time (ms)	23.06	23.33	23.29	25.35	25.35
PSNR (dB)	30.66	30.60	31.10	31.24	31.27

4.2. Omniscient vs Iterative, Recurrent and Hybrid

We compare the same generator network under the iterative, recurrent, hybrid and omniscient frameworks (denoted as IVSR, RVSR, HVSR and OVSR), where only necessary adjustments are made to satisfy the form of input/output of these frameworks. Note that IVSR, RVSR and HVSR only embody one generator network, and thus we have to add the Bicubic amplified frame I_t^{Bic} for residual learning [8, 14]. For LOVSR and GOVSR, we also replace $I_t^{SR_p}$ with I_t^{Bic} in Equation (3), and we set $\alpha = 0$ in Equation (4) for a fair comparison. We integrate the multi-channel PFRBs into these frameworks, where we set 5 PFRBs with the filter number as 64 for IVSR, RVSR and HVSR.

As shown in Figure 4(a) and Table 1, HVSR outperforms IVSR and RVSR a lot in PSNR, which confirms the effec-

Table 2: PSNR (dB) evaluated by adjusting the weight of $I_t^{SR_p}$ through changing α in Equation (4).

Setting	+Bicubic	$\alpha = 0$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 1$
LOVSR	31.24	31.25	31.23	31.26	31.17
GOVSR	31.27	31.28	31.36	31.32	31.30

Table 3: PSNR (dB) evaluated by adjusting the residual blocks in the precursor and successor networks. "4+2" denotes setting 4 residual blocks in Net_p and 2 in Net_s

Setting	0+6	1+5	2+4	3+3	4+2	5+1	6+0
LOVSR	31.12	31.26	31.20	31.23	31.28	31.26	31.24
GOVSR	31.20	31.36	31.40	31.46	31.52	31.47	31.43

tiveness of utilizing a hidden state from the past and a LR frame of the future. Nevertheless, as has been discussed in Section 1 and Section 3.1, HVSR still fails to leverage the hidden states from the present and future. Then, to train our models LOVSR and GOVSR, we first set only 1 PFRB in Net_p and 5 PFRBs in Net_s, and we adjust the filter number as 56. This setting aims to keep the parameters, calculation and time costs of these 5 models almost the same. As shown in Table 1, under similar parameters, LOVSR and GOVSR surpasses HVSR about 0.14 dB and 0.17 dB in PSNR respectively. GOVSR enjoys more advantages because it manages to exploit the global information to reconstruct all the frames in a video sequence. Overall, similar generator networks enjoy such a great advancement under the omniscient framework, which proves its effectiveness undoubtedly.

4.3. Reconstruction Refinement

In Section 4.2, because IVSR, RVSR and HVSR do not own a precursor network, we adopt the Bicubic amplified results to replace the SR frames given by the precursor for OVSR. However, the Bicubic amplification is not optimal for our omniscient framework. Since OVSR consists of a precursor network and a successor network, we naturally consider making the precursor and successor learn the structures and details of frames respectively, and then add them for refinement. Thus, we change α in Equation (4) during training to explore the best option.

Training curves are shown in Figure 4(b) and Figure 4(c), while the specific numbers are demonstrated Table 2. For LOVSR, we find $\alpha = 0.1$ to be the optimal option, which surpasses the "+Bicubic" 0.02 dB. For GOVSR, setting $\alpha = 0.01$ achieves 31.36 dB in PSNR, which surpasses the "+Bicubic" 0.09 dB. We reckon that Net_p and Net_s in GOVSR cooperate better due to the global information utilization.

It is worth mentioning that in some case, especially when imposing no restrictions on the precursor ($\alpha = 0$), as illustrated in Figure 4(b) and Figure 4(c), models could suffer sudden drops during training, but they will still converge eventually.

4.4. Proportion of Precursor and Successor

In Section 4.2 and Section 4.3, we only set 1 PFRB in the precursor and 5 PFRBs in the successor for OVSR. Nevertheless, the optimal proportion of Net_p and Net_s still needs to be explored. Thus, we keep the total number of residual blocks fixed (6), and adjust the residual blocks in Net_p (from 0 to 6) and Net_s (from 6 to 0) to find an appropriate proportion. Training curves are shown in Figure 4(d) and Figure 4(e), while the specific numbers are demonstrated Table 3. Obviously, neither "0+6" or "6+0" is the best or second-best option, which verifies the necessity of designing both the precursor and successor networks. Both LOVSR and GOVSR achieve their best performances with 4 residual blocks in the precursor and 2 in the successor, but GOVSR achieves 31.52 dB under the optimal setting, which again proves the advantage of utilizing global information. Note that "4:2" is the optimal proportion under 6 residual blocks in total, however, there could be a better proportion if with more residual blocks.

After optimization, our models LOVSR and GOVSR achieve 31.28 dB and 31.52 dB in PSNR, which surpass the HVSR 0.18 dB and 0.42 dB respectively. With similar parameters and calculation costs, our OVSR outperforms the existing IVSR, RVSR and HVSR enough to prove the effectiveness of further involving estimated hidden states from the present and future.

4.5. Comparisons with SOTA Methods

Because most methods train their models on different training datasets with different down-sampling kernels, it is unfair to compare these methods only according to their papers. Thus, we have reimplemented dozens of SOTA VSR methods on the same training dataset with the same down-sampling kernels. Since PFNL [29] has already rebuilt DRVSR [22], FRVSR [19] and DUF_52L [12] under the same training conditions, we adopt their public codes. Based on the public code of PFNL, we reimplement MMCNN [25] and MTUDM [30] on the TensorFlow platform. Moreover, We rebuild RBPN [7], EDVR [24], FFCVSR [27], TDAN [23], RSDN [10] and RLSP [5] on the PyTorch platform. We could not retrain TGA because it requires too much GPU memory, and we have to report the results in its paper.

We first reimplement these VSR methods on MM522 [25] training dataset and then test them on Vid4 [1] and UDM10 [30] testing dataset. The PSNR and SSIM values are calculated only on the luminance channel of YCbCr colorspace, skipping the first and last two frames and elim-

Table 4: PSNR (dB) / SSIM of different video SR models on Vid4 testing dataset [1] by the upscaling factor of 4. Red and blue respectively indicate the best and second-best results. The * denotes the results reported in the original papers.

Methods	calendar	city	foliage	walk	average	average*
DRVSR [22]	22.88 / 0.7586	27.06 / 0.7698	25.58 / 0.7307	29.11 / 0.8876	26.16 / 0.7867	25.52 / 0.7600
FRVSR [19]	23.46 / 0.7854	27.70 / 0.8099	25.96 / 0.7560	29.69 / 0.8990	26.70 / 0.8126	26.69 / 0.8220
MMCNN [25]	23.63 / 0.7969	27.47 / 0.8083	26.01 / 0.7532	29.94 / 0.9030	26.76 / 0.8154	26.28 / 0.7844
MTUDM [30]	23.76 / 0.8026	27.67 / 0.8145	26.08 / 0.7587	30.16 / 0.9069	26.92 / 0.8207	26.57 / 0.7989
DUF_52L [12]	23.85 / 0.8052	27.97 / 0.8253	26.22 / 0.7646	30.47 / 0.9118	27.13 / 0.8267	27.34 / 0.8327
RBPN [7]	24.33 / 0.8244	28.28 / 0.8413	26.46 / 0.7753	30.58 / 0.9130	27.41 / 0.8385	27.16 / 0.8190
EDVR [24]	24.30 / 0.8242	28.04 / 0.8382	26.45 / 0.7744	30.63 / 0.9140	27.36 / 0.8377	27.35 / 0.8264
PFNL [29]	24.37 / 0.8246	28.09 / 0.8385	26.51 / 0.7768	30.64 / 0.9134	27.41 / 0.8383	27.40 / 0.8384
FFCVSR [27]	24.39 / 0.8250	27.80 / 0.8314	26.70 / 0.7868	30.55 / 0.9124	27.36 / 0.8389	26.97 / 0.8300
RLSP7-256 [5]	24.60 / 0.8335	28.14 / 0.8453	26.75 / 0.7925	30.88 / 0.9192	27.60 / 0.8476	27.55 / -
TDAN [23]	23.56 / 0.7896	27.53 / 0.8028	26.00 / 0.7491	29.99 / 0.9032	26.77 / 0.8112	26.86 / 0.8140
TGA* [11]	24.47 / 0.8286	28.37 / 0.8419	26.59 / 0.7793	30.96 / 0.9181	27.59 / 0.8419	27.59 / 0.8419
RSDN9-128 [10]	24.74 / 0.8386	28.75 / 0.8554	27.00 / 0.8013	30.85 / 0.9183	27.83 / 0.8534	27.92 / 0.8505
LOVSR-4+2-56 (ours)	24.71 / 0.8378	28.47 / 0.8502	26.94 / 0.7969	30.97 / 0.9204	27.78 / 0.8513	27.78 / 0.8513
LOVSR-8+4-56 (ours)	24.93 / 0.8439	29.08 / 0.8616	27.11 / 0.8073	31.23 / 0.9239	28.09 / 0.8592	28.09 / 0.8592
LOVSR-8+4-80 (ours)	25.10 / 0.8515	28.97 / 0.8666	27.25 / 0.8124	31.47 / 0.9273	28.20 / 0.8644	28.20 / 0.8644
GOVSR-4+2-56 (ours)	24.88 / 0.8463	28.74 / 0.8614	27.06 / 0.8075	31.27 / 0.9245	27.99 / 0.8599	27.99 / 0.8599
GOVSR-8+4-56 (ours)	25.16 / 0.8556	28.76 / 0.8683	27.36 / 0.8190	31.60 / 0.9290	28.22 / 0.8680	28.22 / 0.8680
GOVSR-8+4-80 (ours)	25.28 / 0.8581	29.10 / 0.8769	27.49 / 0.8230	31.79 / 0.9314	28.41 / 0.8724	28.41 / 0.8724

Table 5: PSNR (dB) / SSIM of different video SR models on UDM10 testing dataset [30] by the upscaling factor of 4. Red and blue respectively indicate the best and second-best results.

Methods	archpeople	archwall	auditorium	band	caffe	camera	clap	lake	photography	polyflow	average
DRVSR [22]	35.83/0.9547	41.16/0.9671	29.00/0.9039	34.32/0.9579	39.08/0.9715	45.19/0.9905	36.20/0.9635	31.15/0.8440	36.60/0.9627	37.91/0.9565	36.64/0.9472
FRVSR [19]	36.24/0.9579	41.65/0.9710	29.81/0.9181	34.54/0.9589	39.82/0.9746	46.07/0.9912	36.51/0.9659	31.70/0.8623	36.95/0.9655	38.38/0.9597	37.17/0.9525
MMCNN [25]	36.95/0.9636	42.12/0.9729	30.05/0.9217	35.23/0.9645	40.29/0.9760	46.89/0.9922	37.32/0.9704	31.76/0.8642	37.81/0.9704	38.85/0.9649	37.73/0.9561
MTUDM [30]	37.16/0.9655	42.33/0.9744	30.37/0.9274	35.46/0.9661	40.68/0.9773	47.15/0.9924	37.69/0.9727	32.03/0.8734	38.18/0.9727	39.10/0.9670	38.02/0.9589
DUF_52L [12]	36.92/0.9638	42.53/0.9754	30.27/0.9257	35.49/0.9660	41.03/0.9785	47.30/0.9927	37.70/0.9719	32.06/0.8730	38.02/0.9719	39.25/0.9667	38.05/0.9586
RBPN [7]	38.50/0.9729	43.53/0.9790	31.23/0.9376	35.49/0.9678	41.83/0.9810	49.25/0.9940	38.35/0.9757	32.48/0.8837	38.96/0.9771	40.38/0.9732	39.00/0.9642
EDVR [24]	38.46/0.9732	43.35/0.9783	31.15/0.9372	35.97/0.9696	41.76/0.9808	49.49/0.9947	38.22/0.9759	32.21/0.8790	39.40/0.9793	40.47/0.9739	39.05/0.9642
PFNL [29]	38.35/0.9724	43.55/0.9792	31.18/0.9369	36.01/0.9691	41.84/0.9808	49.26/0.9941	38.33/0.9756	32.53/0.8865	38.95/0.9768	40.04/0.9734	39.00/0.9645
FFCVSR [27]	37.50/0.9667	42.98/0.9766	30.50/0.9270	35.71/0.9669	41.27/0.9798	48.65/0.9936	37.88/0.9727	32.23/0.8729	38.42/0.9739	39.74/0.9691	38.49/0.9599
RLSP7-256 [5]	38.05/0.9704	43.46/0.9787	31.01/0.9342	36.05/0.9693	42.06/0.9818	49.14/0.9939	38.41/0.9756	32.60/0.8865	39.03/0.9771	40.38/0.9748	39.02/0.9642
TDAN [23]	37.95/0.9699	42.60/0.9747	30.54/0.9283	35.23/0.9645	40.59/0.9773	48.38/0.9936	37.42/0.9714	31.87/0.8668	38.28/0.9740	39.00/0.9660	38.19/0.9586
RSDN9-128 [10]	38.36/0.9719	43.68/0.9796	31.65/0.9416	36.13/0.9696	42.22/0.9824	49.88/0.9946	38.48/0.9762	32.67/0.8860	39.47/0.9793	40.44/0.9735	39.30/0.9655
LOVSR-4+2-56 (ours)	38.26/0.9718	43.57/0.9794	31.22/0.9376	36.23/0.9702	42.28/0.9825	49.43/0.9942	38.51/0.9763	32.73/0.8905	39.24/0.9781	40.43/0.9753	39.19/0.9656
LOVSR-8+4-56 (ours)	38.61/0.9736	43.84/0.9804	31.78/0.9437	36.52/0.9717	42.79/0.9836	50.28/0.9948	38.95/0.9781	32.89/0.8932	39.89/0.9807	40.99/0.9773	39.65/0.9677
LOVSR-8+4-80 (ours)	39.01/0.9755	44.13/0.9815	32.24/0.9480	36.80/0.9732	43.16/0.9843	50.52/0.9950	39.26/0.9793	33.13/0.9006	40.29/0.9820	41.32/0.9785	39.99/0.9698
GOVSR-4+2-56 (ours)	38.44/0.9727	43.73/0.9802	31.48/0.9408	36.32/0.9708	42.58/0.9833	49.54/0.9942	38.80/0.9781	32.89/0.8969	39.44/0.9790	40.45/0.9767	39.37/0.9673
GOVSR-8+4-56 (ours)	38.79/0.9747	44.12/0.9818	32.30/0.9488	36.82/0.9734	43.09/0.9841	50.46/0.9949	39.35/0.9802	33.22/0.9044	40.29/0.9821	41.17/0.9791	39.96/0.9703
GOVSR-8+4-80 (ours)	39.01/0.9757	44.44/0.9828	32.53/0.9510	37.03/0.9744	43.07/0.9845	50.60/0.9950	39.58/0.9809	33.34/0.9066	40.48/0.9830	41.28/0.9791	40.14/0.9713

inating 8 pixels on four borders of each frame [13, 29]. As illustrated in Table 4, our heavy model GOVSR-8+4-80 has achieved the best result, while our medium model GOVSR-8+4-56 achieves the second-best performance. LOVSR-based models behave a little worse than GOVSR-based models because they can not utilize the global information, but they still outperform most other SOTA methods.

Because Vid4 dataset contains only 4 scenes at a low resolution, we further test these methods on a bigger testing dataset UDM10. As many methods have not conducted experiments on UDM10 dataset, we only report the results rebuilt by us in Table 5. Our heavy model GOVSR-8+4-80 achieves the best performance, while GOVSR-8+4-56 and LOVSR-8+4-80 achieve comparable performances. To make a comprehensive comparison, we show more details of these methods, *e.g.* type of framework and generator,

Table 6: Comprehensive comparisons of different video SR methods.

Methods	Framework	Generator	Frame	Parameter (M)	FLOPs (T) 1280 × 720	Time (ms) / FPS 1280 × 720	Time (ms) / FPS 1920 × 1080
DRVSR [22]	iterative	recurrent	3	1.722	0.415	123.6 / 8.09	276.2 / 3.62
FRVSR [19]	recurrent	recurrent	2	5.058	0.348	77.5 / 12.90	172.1 / 5.81
MMCNN [25]	iterative	recurrent	5	10.582	3.347	696.7 / 1.44	1548.9 / 0.65
MTUDM [30]	iterative	recurrent	5	5.919	1.672	506.9 / 1.97	1127.6 / 0.89
DUF_52L [12]	iterative	iterative	7	5.824	2.348	1108.0 / 0.90	2499.8 / 0.40
RBPN [7]	iterative	recurrent	7	12.772	8.516	6555.7 / 0.15	14935.8 / 0.07
EDVR [24]	iterative	iterative	7	20.699	2.954	436.7 / 2.29	979.2 / 1.02
PFNL [29]	iterative	iterative	7	3.003	0.940	231.0 / 4.33	567.0 / 1.76
FFCVSR [27]	hybrid	hybrid	3	5.581	0.322	50.8 / 19.69	112.7 / 8.87
RLSP7-256 [5]	hybrid	hybrid	3	5.553	0.320	42.9 / 23.31	91.9 / 10.88
TDAN [23]	iterative	iterative	5	2.285	0.558	102.3 / 9.78	225.4 / 4.44
TGA [11]	iterative	iterative	7	7.058	0.700	383.5 / 2.61	869.7 / 1.15
RSDN9-128 [10]	recurrent	recurrent	2	6.180	0.356	59.3 / 16.86	132.0 / 7.58
OVSR-4+2-56 (ours)	omniscient	omniscient	3	1.897	0.110	25.4 / 39.37	56.1 / 17.83
OVSR-8+4-56 (ours)	omniscient	omniscient	3	3.480	0.201	46.5 / 21.51	101.9 / 9.81
OVSR-8+4-80 (ours)	omniscient	omniscient	3	7.062	0.407	81.2 / 12.32	178.5 / 5.60

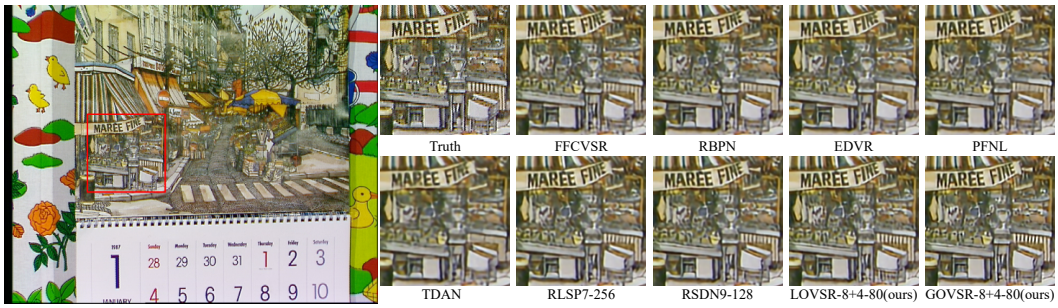


Figure 5: Results of different methods on *calendar* from Vid4 [1] dataset.

frame number, parameters, FLOPs, testing time cost and frames per second (FPS). As demonstrated in Table 6, our light model OVSR-4+2-56 achieves the fastest speed and capable of real-time $4\times$ VSR for 720p. Our medium model OVSR-8+4-56 surpasses all of the other SOTA methods a lot in PSNR and is still faster than almost all of them. Our heavy model OVSR-8+4-80 has to sacrifice the speed for achieving the best performance, which is necessary and worthwhile. As illustrated in Figure 5, our methods can restore clearer details. In all, our methods are superior to these SOTA VSR counterparts in objective metrics, subjective visual effects and complexity.

5. Conclusion

In this paper, we propose an omniscient framework for VSR, which can be divided into the local omniscient VSR and global omniscient VSR. LOVSR suits online VSR and GOVSR fits offline VSR, and they both leverage LR frames and estimated hidden states from the past, present and future. We have conducted abundant experiments to prove the robustness of the omniscient framework, which is superior to other frameworks under similar complexity. Combined with a sophisticated generator network, the omniscient framework makes it possible to conduct VSR in real-time but still with an amazing performance. Due to the overwhelming advantages of this framework, we hope it will become a standard framework in VSR, based on which the researchers are welcome to design more effective generators, motion information capturing modules, as well as loss functions to further unleash its potential.

Acknowledgements

This research is funded by National Natural Science Foundation of China (62071339, U1903214, U1736206, 62072347, 62072350, 61971165, 61971315, 61872277), Hubei Province Technological Innovation Major Project (2019AAA049), Hubei Province Key R&D Project (2020BAB018).

References

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

[2] Kelvin C.K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4947–4956, June 2021.

[3] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11065–11074, June 2019.

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

[5] Dario Fuoli, Shuhang Gu, and Radu Timofte. Efficient video super-resolution through recurrent latent space propagation. In *IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 3476–3485, 2019.

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

[7] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

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

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

[10] Takashi Isobe, Xu Jia, Shuhang Gu, Songjiang Li, Shengjin Wang, and Qi Tian. Video super-resolution with recurrent structure-detail network. In *European Conference on Computer Vision (ECCV)*, pages 645–660, 2020.

[11] Takashi Isobe, Songjiang Li, Xu Jia, Shanxin Yuan, Gregory Slabaugh, Chunjing Xu, Ya-Li Li, Shengjin Wang, and Qi Tian. Video super-resolution with temporal group attention. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8008–8017, June 2020.

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

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

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

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

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

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

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

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

[20] Wenzhe Shi, Jose Caballero, Ferenc Huszr, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

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

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

[23] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. Tdan: Temporally-deformable alignment network for video super-resolution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3360–3369, June 2020.

[24] Xintao Wang, Kelvin C.K. Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1954–1963, June 2019.

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

[26] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.

[27] Bo Yan, Chuming Lin, and Weimin Tan. Frame and feature-context video super-resolution. In *AAAI Conference on Artificial Intelligence*, pages 5597–5604, 2019.

[28] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, Tao Lu, and Jiayi Ma. A progressive fusion generative adversarial network for realistic and consistent video super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[29] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, and Jiayi Ma. Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3106–3115, 2019.

[30] Peng Yi, Zhongyuan Wang, Kui Jiang, Zhenfeng Shao, and Jiayi Ma. Multi-temporal ultra dense memory network for video super-resolution. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(8):2503–2516, 2020.

[31] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *The European Conference on Computer Vision (ECCV)*, pages 294–310, September 2018.

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