Propagating Difference Flows for Efficient Video Super-Resolution

Ruisheng Gao
grsmc4180@mail.ustc.edu.cn
Zeyu Xiao
zeyuxiao@mail.ustc.edu.cn
Zhiwei Xiong ✉
zwxiong@ustc.edu.cn

Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei, China

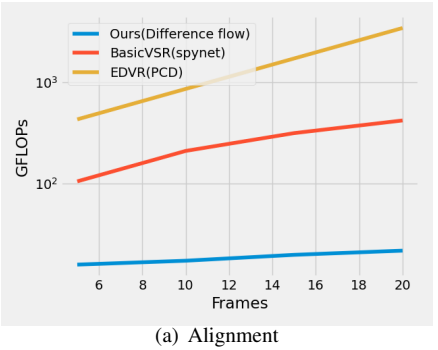
Abstract

Recent years have witnessed the advancement of video super-resolution (VSR) with elaborately-designed multi-frame alignment and space-time fusion/refinement techniques. However, both techniques require heavy computational burden and memory consumption, hindering existing VSR networks from being deployed on resource-constrained platforms (*e.g.*, smartphones and wearable devices). In this paper, we propose an efficient and lightweight VSR network with two special designs. First, we propose a novel motion propagation scheme which propagates *difference flows* for efficient feature alignment. The difference flow is sparse and computational-friendly which focuses on texture details. After estimating the preliminary difference flow with an initial motion estimator, we then design an adaptive motion modification module for frame-pair wise adaptation through bidirectional propagation. Second, a dense feature distillation module is designed for further refining the aligned features efficiently. Thanks to both designs, our network achieves comparable performance with state-of-the-art VSR methods while enjoying a clear advantage in model size and computational efficiency.

1 Introduction

Video super-resolution (VSR) aims to generate a high-resolution (HR) video from its corresponding low-resolution (LR) observation. HR videos are desired in various applications, such as video surveillance, high-definition television and video streaming. Recently, deep learning based method have demonstrated promising performance for VSR [3, 5, 15, 19, 22, 32, 36, 42], and the state-of-the-art methods outperform traditional methods [2, 8, 21] with notable gains. Such performance boost is obtained by various complex temporal modeling techniques, *i.e.*, implicit/explicit multi-frame alignment and space-time fusion/refinement, which hinder the deployment of VSR networks on resource-constrained platforms, *e.g.*, smartphones and wearable devices.

To reduce the required computational cost and memory consumption, we argue that, designing efficient alignment and multi-frame fusion/refinement schemes are the keys to lightweight VSR networks.



(a) Alignment (b) Fusion/refinement and reconstruction

Figure 1: Complexity analysis (GFLOPs) of typical VSR networks (*i.e.*, EDVR [36] and BasicVSR [3]). The value of GFLOPs is calculated on super-resolving a video clip of spatial resolution 128×256 with the scale factor equal to 4. (a) Comparison between different multi-frame alignment schemes. (b) Comparison between different feature fusion/refinement and reconstruction schemes. Note that, due to different network designs, the computation cost of feature fusion/refinement and reconstruction are summed for a fair comparison.

(1) Alignment. A core step in VSR is to align different frames, either explicitly or implicitly. It is especially worth mentioning that, as an advanced solution for VSR, BasicVSR [3] utilizes the pretrained SpyNet [28] for optical flow estimation first and then warps each frame feature bidirectionally for explicit alignment. However, estimating optical flow itself is time-consuming and introduces extra parameters (*e.g.*, Spynet introduces 1.4M parameters). Furthermore, as can be seen in Fig. 1(a), with the increased number of input frames, the computational cost in terms of GFLOPs grows linearly, indicating an explicit alignment scheme is not affordable in resource-constrained applications. To avoid explicitly calculating optical flow, some recent methods exploit the motion information in an implicit manner. For example, EDVR [36] utilizes the pyramid, cascading and deformable (PCD) alignment module for multi-frame alignment and shows superior performance under large motions due to the flexible receptive fields. However, such implicit scheme still suffers from heavy computational cost, as shown in Fig. 1(a).

(2) Fusion/refinement. Another resource-consuming part comes from the deep stacked backbones, among which residual blocks play an important role in the feature fusion/refinement and reconstruction processes. For example, BasicVSR [3] fuses the aligned features with 30 residual blocks that consists of 60 convolutional layers, and EDVR [36] utilizes 40 residual blocks in the reconstruction process that costs over 4,000 GFLOPs to generate a 5-frame clip, as can be seen in Fig. 1(b).

Therefore, more efficient alignment methods as well as lighter fusion/refinement structures are desired to enable efficient VSR in resource-constrained applications. In this paper, we attempt to reduce computational cost and model size from the aforementioned two aspects and propose a novel network for efficient VSR. For the former part, we propose a motion propagation scheme which is different from existing methods. With an initial motion estimator, our network estimates motion fields only in the beginning and the end of an input sequence, which gets rid of dense flow estimation per frame pair. The estimated preliminary "difference flows" are then propagated along the whole sequence in a bidirectional manner, which are modified at each timestep through an adaptive motion modification module to fit motion changes. Instead of estimating optical flow with extra pretrained networks, our motion estimation is based on the temporal difference, which is lightweight and

efficient compared with previous methods [3, 40] (see Fig. 1(a)). On the other hand, the residual feature distillation block (RFDB) originally proposed in [23] is extended to a dense alternative for effective feature refinement in each propagating branch. The proposed dense feature distillation (DFD) module can efficiently refines the warped features by distilling useful channels over distilled features without deep stacked structures (see Fig. 1(b)).

The contributions of this paper are summarized as follows. (1) A novel motion propagation scheme is proposed for efficient feature alignment, which estimates preliminary motion fields called "difference flow" and modifies them per frame pair adaptively. (2) A dense alternative of RFDB is designed to distill and refine warped features without deep stacked structures, which maintains the efficiency and effectiveness of the network. (3) Experimental results demonstrate that the proposed network achieves comparable performance with state-of-the-art VSR methods on two benchmark datasets Vid4 and Vimeo90K-T while enjoying a clear advantage in model size and computational efficiency.

2 Related Work

Single Image Super-Resolution. Single image super-resolution (SISR) research has been flourishing since its first attempt on deep learning [7]. The performance of SR results keep reaching new peaks through deeper networks [14, 20, 31], attention mechanisms [41, 43] and vision Transformers [18]. Advanced augmentation techniques are also proposed which further boost current methods [35]. However, deeper structures as well as fancy but complex modules inevitably bring computational burdens. Therefore, designing lightweight and efficient networks becomes a promising topic in SISR [12, 23, 26, 39]. For example, Hui *et al*. [12] devise a lightweight information multi-distillation block for hierarchical feature extraction and adaptive fusion, followed by the contrast-aware channel attention mechanism to emphasize important features. Liu *et al*. [23] use a simple convolutional layer to replace the channel splitting operation and propose the RFDB to save the inference time. In this paper, motivated by [23], we propose a dense alternative of RFDB, which further distills useful features for feature refinement.

Video Super-Resolution. VSR advances SISR by exploiting additional information from neighboring low-resolution frames. Most VSR methods embed additional optical flow estimation networks to calculate accurate pixel-wise dense motion field in order to align multiple frames [3, 19, 29, 37, 38, 40]. For example, Chan *et al*. [3] align multiple adjacent frames in the feature level using the pretrained SpyNet [28] and propose a bidirectional network for long-term temporal modeling. To avoid explicitly calculating the optical flow, some methods exploit the motion information in an implicit manner, among which the deformable convolution [6] receives much attention due to its flexible receptive field and is utilized to align frames in the feature domain in VSR [32, 36]. However, even though the unstable training phenomenon is alleviated by [4], heavy computational burdens are still hard to overcome. Different from existing VSR methods, our proposed network does not need extra parameters (*i.e.*, pretrained flow estimation network) for motion estimation. With an initial motion estimator, the estimated difference flows are propagated along the whole sequence for efficient feature alignment.

Temporal Difference for Motion Modeling. Being widely used in video action recognition tasks, temporal difference serves as an efficient and effective motion estimation operator [24, 33, 34]. However, its potential in VSR has rarely been explored. Recently, Isobe *et al*. [13] utilize explicit image level difference in both LR and HR spaces for motion modeling. In

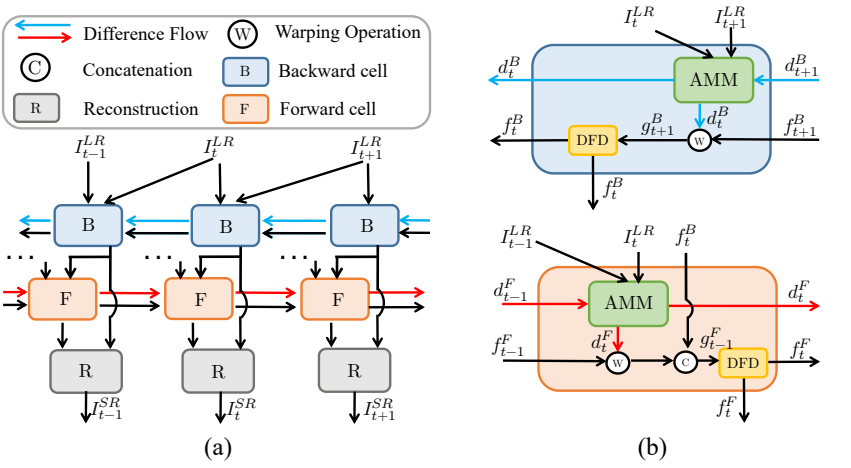


Figure 2: An overview of the proposed network. (a) shows the bidirectional propagation of motions and features by our method. B, F and R denote the backward cell, the forward cell and the reconstruction module. Input frames of the forward cells are substituted by ellipsis for clearer illustration, which are the same as backward cells. (b) shows the details of the forward and backward cells, which include an adaptive motion modification (AMM) module and a dense feature distillation (DFD) module. $d_t^{F/B}$ denotes the motion (forward/backward difference flow) propagated at time step t. $f_t^{F/B}$ denotes the feature propagated at time step t. The motion propagated from the next/previous time step is modified by the AMM module for feature warping and the warped feature is then refined by the DFD module.

contrast, our method combines image level and feature level differences to estimate and propagate sparse difference flows, which requires little extra computation.

3 Method

3.1 Overview

Given an LR sequence $\{I_t^{LR}\}_{t=1}^N$ consisting of N frames, where $I_t^{LR} \in \mathbb{R}^{3\times H\times W}$, VSR aims to recover its HR counterpart $\{I_t^{SR}\}_{t=1}^N$, which should be close to the ground truth $\{I_t^{HR}\}_{t=1}^N$, where $I_t^{SR/HR} \in \mathbb{R}^{3\times sH\times sW}$. H, W and s denote the height, width and the scale factor, respectively. The overall pipeline of our method is shown in Fig. 2.

Different from sliding window-based methods (*e.g.*, EDVR [36]), we follow a typical bidirectional propagation scheme [3] to propagate both features and estimated difference flows. Specifically, we design an Initial Motion Estimator (IME) to obtain a preliminary forward (backward) flow using the first (last) two frames of the input sequence. Then the estimated bidirectional preliminary flows are propagated and modified by the Adaptive Motion Modification (AMM) module for frame-pair wise adaptation in each time step. For a given time step t, the reference frame I_t^{LR} utilizes the modified flow for feature alignment through the warping operation. Afterwards, the aligned features are fed into the Dense Feature Distillation (DFD) module for further refinement. Finally, the upsampled reference frame is obtained by passing several convolutional layers and the pixel-shuffle [30] operation, which follows the design of BasicVSR [3].

3.2 Propagating Difference Flows

Existing VSR methods deal with multi-frame alignment by using either a pretrained flow estimation network followed by the warping operation [3] or learnable offsets in the feature domain (*i.e.*, deformable convolution [36] and attention map [1]). Unlike the above two alignment schemes that require large computational cost, we propose to propagate the motion estimated at the beginning (end) of the input sequence for efficient multi-frame alignment. Our motion propagation scheme consists of two modules: the IME module and the AMM module. Without loss of generality, we take the backward branch as an example in this subsection and the forward branch can be similarly derived. Note that the forward cell includes an extra concatenation operation.

Initial Motion Estimator (IME). The main motivation of flow propagation here is to avoid dense motion prediction between each frame pair. Intuitively, assuming continuous motion between video frames, we can obtain a rough motion estimation in the current time step based on its counterpart in the last time step. We first estimate the motion between the last two frames. As shown in Fig. 3(a), we first extract differences at both image and feature levels. Here we use a shared convolutional layer to extract features of input frames for feature level difference calculation. Then, we concatenate the original input frames to utilize frame wise temporal information. To take advantage of long-range spatial dependencies, we further use the Large Kernel Attention (LKA) [9] to enlarge the receptive field of the above three parts for fine-grained motion estimation. Finally, the output of LKA layers are fused to form the preliminary motion field d^B_{N-1} of the backward branch. This process can be formulated as

$$d^B_{N-1} = F(Concat(\varphi_1(r^i_{N,N-1}), \varphi_2(r^f_{N,N-1}), \varphi_3(f_{N,N-1}))), \tag{1}$$

where $r^i_{N,N-1}$, $r^f_{N,N-1}$ denotes the image and feature level differences of the N-th (the last) frame and its predecessor frame, respectively. $f_{N,N-1}$ denotes frame features of the last two frames. $\varphi_j(.)(j = 1,2,3)$ and $F(.)$ stand for the LKA layer and the fusion operation, respectively.

Adaptive Motion Modification (AMM). After estimating the preliminary motion in both the forward and the backward directions, we need to modify the motion in each time step for adaptive motion modeling. Concerning about the complexity, we devise a simple module based on image level difference to tailor frame-pair wise motion information, as can be seen in Fig. 3(b). The motion estimated from the next time step is first concatenated with the image level difference between the current reference frame I^{LR}_t and its successor I^{LR}_{t+1}, following by the stacked 1×1 and 3×3 convolutional layers for channel-wise and spatial-wise adaptation. Formally, we can derive the modified motion as

$$d^B_t = \phi(Concat(d^B_{t+1}, r^i_{t+1,t})), \tag{2}$$

where $\phi(.)$ denotes the stacked 1×1 and 3×3 convolutional layers. We call the estimated and propagated motion field "difference flow", which benefits from the difference operation. Detailed analysis of the difference flow is shown in Sec. 4.3. After obtaining the modified flow, a warping operation is conducted for feature alignment, as can be seen in Fig. 2. The aligned features are fed into the DFD module for further refinement and fusion in a computational-friendly manner.

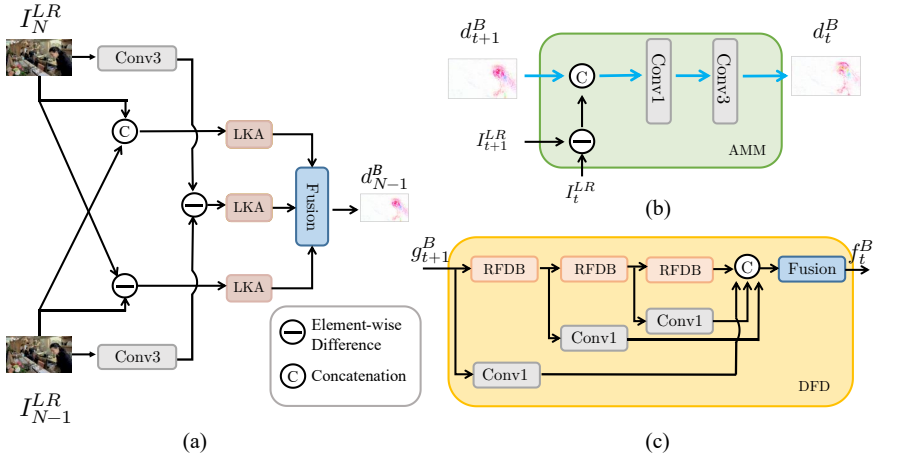


Figure 3: Details of the core modules (backward propagation as an example). (a) The IME module which utilizes image and feature level residual to estimate the preliminary difference flow. LKA denotes the large kernel attention layer [9]. (b) The AMM module which adaptively modifies difference flow propagated from the adjacent time step. (c) The DFD module which uses 1x1 convolutions to densely impose further distillation. RFDB denotes the residual feature distillation block [23].

3.3 Dense Feature Distillation (DFD)

Most existing VSR methods select deep stacked residual blocks as the backbone network (*e.g.*, at least 30 residual blocks, which include over 60 3×3 convolutional layers), which demonstrate promising results due to the residual connections [11] and deep architecture. Naturally, deeper structures introduce heavy computational burdens, which may not be affordable for resource-constrained platforms.

Recently, channel splitting (CS) operation has been widely adopted in efficient SISR [12, 23]. By dividing the input features into a retained one and a coarser one, redundant computation on useless features is saved. Furthermore, cascaded CS operations distill the input features step by step, gradually generating compact representations. As an efficient alternative, RFDB replaces the CS operation with 1×1 convolutional layers which greatly reduces the inference time. Inspired by the efficient design [23], we attempt to use the RFDB as the backbone for feature refinement. However, directly stacking multiple RFDBs shows inferior performance as can be seen in Table 3. To address the above issue, we design a dense alternative of the RFDB, *i.e.*, the DFD module, as can be seen in Fig. 3(c). Instead of sequentially stacking RFDBs to deepen the network, we use 1×1 convolutional layers by densely connecting them with the RFDB layers to further distill useful features. Then the further distilled features are concatenated along the channel dimension and fused by an additional 1×1 convolutional layer. In this way, the input feature benefits from a more complete distillation process with multiple interactions among the outputs of intermediate RFDBs. The effectiveness of the DFD module is demonstrated in Sec. 4.3.

3.4 Training Objective

We impose two loss terms to supervise both intermediate motion fields and the final reconstruction results, named $\mathcal{L}_{\mathrm{M}}$ and $\mathcal{L}_{\mathrm{R}}$, respectively. For the reconstruction loss, we choose

Charbonnier loss [17], which handles outliers much better than the conventional L1 penalty

$$\mathcal{L}_{\mathrm{R}}(I^{HR}, I^{SR}) = \sum_{i=1}^{N} \sqrt{\left\| I_i^{HR} - I_i^{SR} \right\|^2 + \varepsilon^2}. \tag{3}$$

As for intermediate motion constraint, we impose the warping loss in both forward and backward branches, which uses estimated motion fields to warp input LR frames and aligns them with their neighboring frames. Then, the Charbonnier loss is utilized again on the warped LR images and the original ones. This process can be formulated as follows

$$\mathcal{L}_{\mathrm{M}}(I^{LR}, d^F, d^B) = \lambda_1 \sum_F \mathcal{L}_{\mathrm{R}}(\mathrm{W}(I^{LR}, d^F), I^{LR}) + \lambda_2 \sum_B \mathcal{L}_{\mathrm{R}}(\mathrm{W}(I^{LR}, d^B), I^{LR}), \tag{4}$$

where W refers to the warping operation, F and B are denoted as the forward and backward processes. Frame indexes are omitted here for clarity. We empirically set the weights λ_1 and λ_2 as 0.5. Then the total loss is the sum of the above two terms

$$\mathcal{L}_{\mathrm{Total}} = \mathcal{L}_{\mathrm{M}} + \mathcal{L}_{\mathrm{R}}. \tag{5}$$

4 Experiments

4.1 Experimental Settings

We test on 4× SR with the bicubic downsampling degradation. Vimeo90K [40] is used as our training dataset. Vid4 [21] and Vimeo90K-T [40] are used as test datasets. We choose Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) to evaluate reconstruction quality. We use the sequence flipping strategy as in [3] to train our network on 14 input frames using the Adam [16] optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.99$). Random horizontal flipping as well as rotation are also used for training data augmentation. The Cosine Annealing scheme [25] is adopted and the initial learning rate is set to 2e-4. We set the batch size to 8 and the patch size to 64×64. Our network is trained for 200K iterations and finetuned using L2 loss for another 200K iterations to improve the overall performance. All the experiments are conducted using PyTorch [27] on NVIDIA RTX 3090 GPUs.

Methods	Params (M)	GFLOPs	Vid4	Vimeo90K-T
Bicubic	-	-	23.78/0.6347	31.32/0.8684
RBPN [10]	12.2	28230.68	27.12/0.8180	37.07/0.9435
EDVR [36]	20.6	2296.50	27.35/0.8264	37.61/0.9489
BasicVSR [3]	6.3	417.82	27.24/0.8251	37.18/0.9450
IconVSR [3]	8.7	579.36	27.39/0.8279	37.47/0.9476
PFNL [42]	3.0	1067.72	26.73/0.8029	36.14/0.9363
EDVR-M [36]	3.3	526.59	27.10/0.8186	37.09/0.9446
BasicVSR-M	3.2	209.46	26.97/0.8145	36.65/0.9404
Ours	3.0	221.15	27.26/0.8230	36.95/0.9428

Table 1: Quantitative results on Vid4 and Vimeo90K-T. Red and blue colors indicate the best and the second-best efficiency/performance of lightweight models, respectively. PSNR/SSIM are calculated on the Y channel. GFLOPs are computed for generating an HR frame with resolution 512 × 1024.

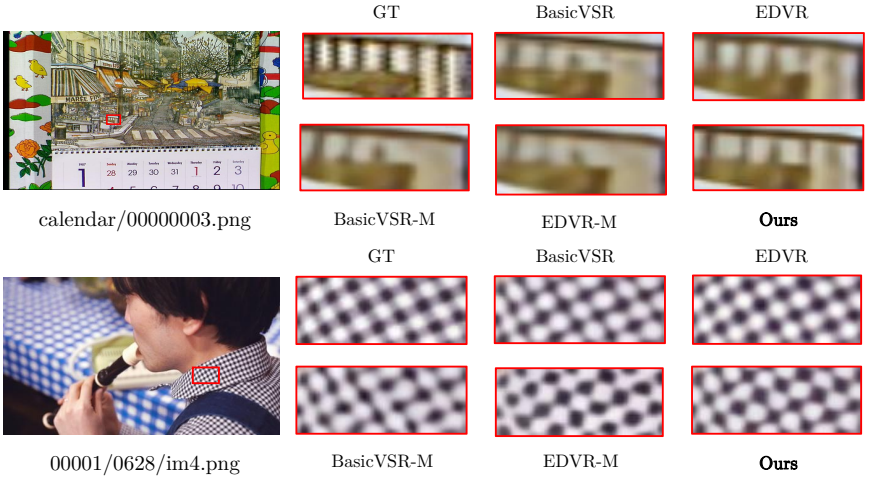


Figure 4: Qualitative results on exemplar scenes of Vid4 (top) and Vimeo90K-T (bottom).

4.2 Comparisons with State-of-the-Art Methods

We compare our network with seven state-of-the-art methods which can be categorized into heavy and lightweight on two widely used benchmarks: Vid4 and Vimeo90K-T. The quantitative results are summarized in Table 1. The performance of most methods is obtained from their original papers for fair comparisons. Note that we decrease the model size of BasicVSR and train a lighter variant BasicVSR-M, which has similar model complexity to ours.

We first compare our method with lightweight models ($\sim$ 3M), including PFNL [42], EDVR-M [36], and BasicVSR-M [3]. As shown in Table 1, our method performs best against the above models on Vid4 with competitive computation cost. For example, we achieve 0.16dB PSNR gain with less than a half GFLOPs compared with EDVR-M, and 0.29dB gain with similar GFLOPs compared with BasicVSR-M. As for Vimeo90K-T, our method outperforms PFNL and BasicVSR-M by 0.81dB and 0.30dB, respectively. When comparing with heavy models such as EDVR, our method has a performance drop but enjoys much superior efficiency on generating multiple frames, as shown in Fig. 1. This advantage enables our method to process longer video sequences with less computational cost, which is essential in resource-constrained applications. We also show some qualitative results on Vid4 and Vimeo90K-T. In the top of Fig. 4, our network is able to produce clearer structures of the stripe pattern compared with other models. In the meantime, as can be seen in the bottom of Fig. 4, our network successfully recovers the grid pattern on the collar, whereas BasicVSR-M and EDVR-M generate severe twists.

4.3 Model Analysis

In order to get deeper insights into the flow propagation scheme, we first visualize the difference flow to make a comparison with the optical flow. We then perform several ablation studies on the core modules in our network on Vid4.

Difference Flow Visualization. To explore the behavior of difference flows, we visualize them from two perspectives. Note that we choose forward flows of both optical flow and difference flow as an example. As shown in the top of Fig. 5, we first compare with optical flow generated by the pretrained SpyNet. Unlike the dense optical flows, the learned difference flows are task-specific, which focus on texture details due to image and feature

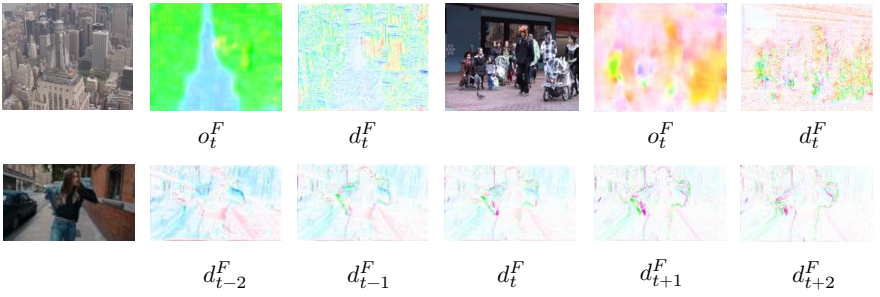


Figure 5: Visualization of difference flows. Top: Comparison between optical flow (o_t^F) and difference flow (d_t^F). Bottom: Difference flows generated on consecutive frames.

Method	PSNR
IME w/o image residual	27.13
IME w/o feature residual	27.12
AMM w/ concat	27.10
w/o AMM	27.16
Spynet	27.20
Ours	27.26

Table 2: Ablation on IME and AMM modules.

Method	#Blocks	PSNR	Params	GFLOPs
Resblock	15	27.26	2.94M	249.04
RFDB	8	27.15	2.94M	223.14
DFD(Ours)	6	27.26	2.95M	221.15

Table 3: Comparison of different feature refinement structures. GFLOPs are computed for generating an 512×1024 output frame.

level differences. Both optical flows and difference flows share similar directions, indicating the capability to represent motion information of the latter. On the other hand, we visualize consecutive difference flows in the bottom of Fig. 5. As the flow propagates through time, it becomes sparser but adaptively keeps in specific region where motion changes. This property makes it possible for efficient alignment without estimating pixel-wise dense flows, which gets rid of extra computation.

Ablation Studies. We perform ablation studies on Vid4 to evaluate the effectiveness of IME, AMM, and DFD modules. To verify the importance of image/feature level residual which serves as motion priors, we discard the image/feature level residual in the IME module. As shown in Table 2, 0.13/0.14dB PSNR drop is observed, which demonstrates the effectiveness of residual information on motion modeling. As for the AMM module, we set three alternatives compared to the original design: (1) AMM w/concat: we directly concatenate two adjacent frames without calculating their frame residual. (2) w/o AMM: we remove the AMM module but keep the IME module. (3) SpyNet: we directly use the pretrained SpyNet to estimate dense optical flows for warping. Being adaptively propagated and modified, difference flows serve as alternative motion cues, making our method outperforms other variants. For evaluating the DFD module, we keep the network structure and training settings but replace the DFD module with stacked residual blocks or sequential RFDBs, respectively. From Table 3 we can see that, while achieving similar performance to stacked residual blocks, DFD costs less computation with shallower design. On the other hand, simply stacking RFDBs leads to 0.11dB PSNR drop, which validates the effectiveness of the DFD module.

Tradeoff between lighter variants. We additionally train different variants of our method by changing the number of channels and the complexity of DFD (40c6b refers to 40 channels and DFD module with 6 RFDBs), resulting in a PSNR-GFLOPs curve in Fig. 6. With the increase of GFLOPs, the performance keeps growing steadily. Note that there is a significant improvement between 1.8M and 2.3M variants, which indicates that finer distillation process contributes to better results.

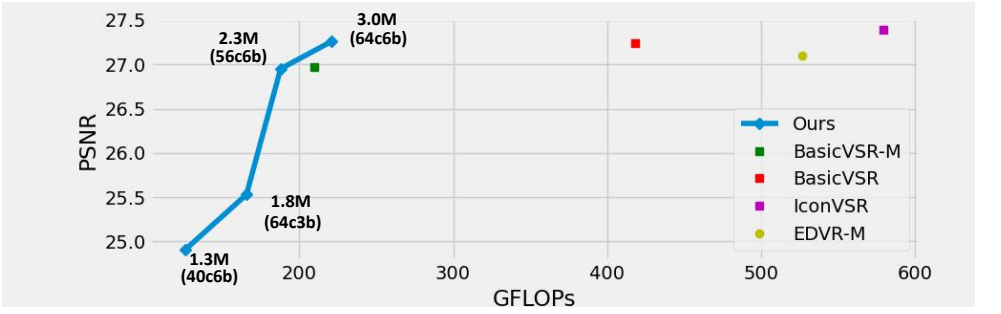


Figure 6: PSNR-GFLOPs tradeoff of our method between lighter variants. We also include several efficient methods for comparison.

5 Conclusion

In this paper, we propose a lightweight and efficient network for VSR. Different from existing VSR methods requiring huge computational cost during multi-frame alignment, we propose a novel motion propagation scheme to propagate the estimated bidirectional difference flow for efficient alignment in the feature domain. On the other hand, we replace commonly used deep stacked residual blocks with an efficient alternative of the residual feature distillation block, which enables multiple interactions among the distilled features. Experimental results on two benchmark datasets demonstrate that our network has comparable performance with several state-of-the-art methods while enjoying a clear advantage in resource-constrained applications.

6 Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62131003 and 62021001.

References

[1] Davide Abati, Amir Ghodrati, Amirhossein Habibian, and Qualcomm AI Research. Efficient video super resolution by gated local self attention.

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

[3] Kelvin CK Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond. In *CVPR*, pages 4947–4956, 2021.

[4] Kelvin CK Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Understanding deformable alignment in video super-resolution. In *AAAI*, 2021.

[5] Zeyuan Chen, Yinbo Chen, Jingwen Liu, Xingqian Xu, Vidit Goel, Zhangyang Wang, Humphrey Shi, and Xiaolong Wang. Videoinr: Learning video implicit neural representation for continuous space-time super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2047–2057, 2022.

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, pages 764–773, 2017.

[7] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[8] Sina Farsiu, M Dirk Robinson, Michael Elad, and Peyman Milanfar. Fast and robust multiframe super resolution. *IEEE transactions on image processing*, 13(10):1327–1344, 2004.

[9] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022.

[10] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. In *CVPR*, pages 3897–3906, 2019.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[12] Zheng Hui, Xinbo Gao, Yunchu Yang, and Xiumei Wang. Lightweight image super-resolution with information multi-distillation network. In *ACM MM*, pages 2024–2032, 2019.

[13] Takashi Isobe, Xu Jia, Xin Tao, Changlin Li, Ruihuang Li, Yongjie Shi, Jing Mu, Huchuan Lu, and Yu-Wing Tai. Look back and forth: Video super-resolution with explicit temporal difference modeling. In *CVPR*, pages 17411–17420, 2022.

[14] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, pages 1646–1654, 2016.

[15] Soo Ye Kim, Jeongyeon Lim, Taeyoung Na, and Munchurl Kim. 3dsrnet: Video super-resolution using 3d convolutional neural networks. *arXiv preprint arXiv:1812.09079*, 2018.

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[17] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, pages 624–632, 2017.

[18] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *ICCV*, pages 1833–1844, 2021.

[19] Jingyun Liang, Jiezhang Cao, Yuchen Fan, Kai Zhang, Rakesh Ranjan, Yawei Li, Radu Timofte, and Luc Van Gool. Vrt: A video restoration transformer. *arXiv preprint arXiv:2201.12288*, 2022.

[20] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR workshops*, pages 136–144, 2017.

[21] Ce Liu and Deqing Sun. On bayesian adaptive video super resolution. *IEEE transactions on pattern analysis and machine intelligence*, 36(2):346–360, 2013.

[22] Chengxu Liu, Huan Yang, Jianlong Fu, and Xueming Qian. Learning trajectory-aware transformer for video super-resolution. In *CVPR*, pages 5687–5696, 2022.

[23] Jie Liu, Jie Tang, and Gangshan Wu. Residual feature distillation network for lightweight image super-resolution. In *ECCV*, pages 41–55. Springer, 2020.

[24] Z Liu, D Luo, Y Wang, L Wang, Y Tai, C Wang, J Li, F Huang, and T Teinet Lu. Towards an efficient architecture for video recognition. In *AAAI*, pages 7–12, 2020.

[25] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. 2016.

[26] Xiaotong Luo, Yuan Xie, Yulun Zhang, Yanyun Qu, Cuihua Li, and Yun Fu. Latticenet: Towards lightweight image super-resolution with lattice block. In *ECCV*, pages 272–289. Springer, 2020.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019.

[28] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, pages 4161–4170, 2017.

[29] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *CVPR*, pages 6626–6634, 2018.

[30] Wenzhe Shi, Jose Caballero, Ferenc Huszár, 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 *CVPR*, pages 1874–1883, 2016.

[31] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *CVPR*, pages 3147–3155, 2017.

[32] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. Tdan: Temporally-deformable alignment network for video super-resolution. In *CVPR*, pages 3360–3369, 2020.

[33] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, pages 20–36. Springer, 2016.

[34] Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *CVPR*, pages 1895–1904, 2021.

[35] Shizun Wang, Ming Lu, Kaixin Chen, Jiaming Liu, Xiaoqi Li, and Ming Wu. Samplingaug: On the importance of patch sampling augmentation for single image super-resolution. 2021.

[36] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *CVPR Workshops*, pages 0–0, 2019.

[37] Zeyu Xiao, Zhiwei Xiong, Xueyang Fu, Dong Liu, and Zheng-Jun Zha. Space-time video super-resolution using temporal profiles. In *ACM MM*, pages 664–672, 2020.

[38] Zeyu Xiao, Xueyang Fu, Jie Huang, Zhen Cheng, and Zhiwei Xiong. Space-time distillation for video super-resolution. In *CVPR*, pages 2113–2122, 2021.

[39] Wenbin Xie, Dehua Song, Chang Xu, Chunjing Xu, Hui Zhang, and Yunhe Wang. Learning frequency-aware dynamic network for efficient super-resolution. In *ICCV*, pages 4308–4317, 2021.

[40] 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.

[41] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *CVPR*, pages 5791–5800, 2020.

[42] 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 *ICCV*, pages 3106–3115, 2019.

[43] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, pages 286–301, 2018.