Kernel Dimension Matters: to Activate Available Kernels for Real-time Video Super-Resolution

Shuo Jin
kingsure@bjtu.edu.cn
Institute of Information Science,
Beijing Jiaotong University
Beijing Key Laboratory of Advanced Information Science and Network Technology
Beijing, China

Meiqin Liu*
mqliu@bjtu.edu.cn
Institute of Information Science,
Beijing Jiaotong University
Beijing Key Laboratory of Advanced Information Science and Network Technology
Beijing, China

Chao Yao
yaochao@ustb.edu.cn
School of Computer and Communication Engineering
University of Science and Technology Beijing
Beijing, China

Chunyu Lin
cylin@bjtu.edu.cn
Institute of Information Science,
Beijing Jiaotong University
Beijing Key Laboratory of Advanced Information Science and Network Technology
Beijing, China

Yao Zhao
yzhao@bjtu.edu.cn
Institute of Information Science,
Beijing Jiaotong University
Beijing Key Laboratory of Advanced Information Science and Network Technology
Beijing, China

ABSTRACT

Real-time video super-resolution requires low latency with high-quality reconstruction. Existing methods mostly use pruning schemes or neglect complicated modules to reduce the calculation complexity. However, the video contains large amounts of temporal redundancies due to the inter-frame correlation, which is rarely investigated in existing methods. The static and dynamic information lies in feature maps and represents the redundant complements and temporal offsets respectively. It is crucial to split channels with dynamic and static information for efficient processing. Thus, this paper proposes a kernel-split strategy to activate available kernels for real-time inference. This strategy focuses on the dimensions of convolutional kernels, including the channel and depth dimensions. Available kernel dimensions are activated according to the split of high-value and low-value channels. Specifically, a multi-channel selection unit is designed to discriminate the importance of channels and filter the high-value channels hierarchically. At each hierarchy, low-dimensional convolutional kernels are activated to reuse the low-value channel and re-parameterized convolutional kernels are employed on the high-value channel to merge the depth dimension. In addition, we design a multiple flow deformable alignment module for a sufficient temporal representation with affordable calculation cost. Experimental results demonstrate that our method outperforms other state-of-the-art (SOTA) ones in terms of reconstruction quality and runtime.

CCS CONCEPTS

• **Computing methodologies** → **Computational photography**.

KEYWORDS

video super-resolution, real-time network, re-parameterization, kernel split

ACM Reference Format:
Shuo Jin, Meiqin Liu, Chao Yao, Chunyu Lin, and Yao Zhao. 2023. Kernel Dimension Matters: to Activate Available Kernels for Real-time Video Super-Resolution. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3611908

1 INTRODUCTION

Since the low-resolution and degradation factors affect video quality, super-resolution plays a vital role in improving video fidelity. Plain video super-resolution (VSR) methods [24, 29] employ deep networks and complicated modules for video feature extraction and reconstruction. Nevertheless, these methods are too heavy and complex to be deployed on devices, where a real-time inference speed should be satisfied. Real-time video super-resolution (RTVSR) aims at restoring the high-resolution (HR) video from its low-resolution (LR) version with a real-time processing pipeline. Compared to plain VSR methods, RTVSR needs to remove the model redundancy and explore efficient modules for real-time inference. Therefore, lightweight feature extractors and efficient motion exploitation modules are key factors to RTVSR.

Existing VSR methods [4, 8] have demonstrated outstanding performance on offline videos. From the perspective of the framework

*Corresponding author.

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3611908

architecture, these methods can be categorized as window-based networks with the multi-input and single-output paradigm and recurrent-based networks with the multi-input and multi-output paradigm. The window-based methods [12, 25] restore the middle frame using several adjacent frames in a short temporal window. Due to the limitation of the window size, window-based methods suffer from a narrow temporal scope and can't leverage the information outside the window. To capture the long-term dependencies, recurrent-based methods employ the unidirectional frameworks [7, 21] to utilize the propagated information from distant frames. A bidirectional recurrent network [3] is further designed with several effective components for information refill and exploits the bidirectional temporal information. Moreover, several methods utilize the non-local mechanism [31] and Transformer [15] to exploit the global temporal information of the video. However, both window-based and recurrent-based methods tend to focus on the repetitive semantic information during the iterative process and feature extraction. In addition, extracting the global temporal information requires a significant amount of memory and computational cost. Consequently, these methods are too heavy and time-consuming to perform real-time inference.

To meet the real-time processing requirement, existing methods explore efficient propagation strategies for fast processing. Dario *et al.* [6] proposes an efficient latent space propagation for iterative transformation. Isobe *et al.* [9] presents a recurrent identity mapping operation to preserve long-range information. Some methods design lightweight alignment strategies to utilize temporal information. Bare *et al.* [1] proposes a lightweight convolutional kernel for alignment and introduced a gated unit with learnable parameters for inter-frame information. However, these methods lack temporal information and achieve limited reconstruction quality gains. Furthermore, some methods employ the pruning scheme [13] on the recurrent-based framework to decrease the time complexity. Zhang *et al.* [34] proposes a sparsity-based learning scheme using a weight normalization layer on the scale parameter to process each frame. Xia *et al.* [26] designs a pruning scheme to learn the structural sparsity of the residual connection. These methods accelerate the runtime, whereas the removed internal parameters lead to a performance drop. Furthermore, few studies focus on the redundancy of static information from feature maps, which is essential for the degree of parallelism at the inference stage.

Figure 1 illustrates the visualization of low-value and high-value channels. It's seen that low-value channels include minimal contexts and tend to supplement static information, while high-value ones contain clear structures that contribute to dynamic information. To eliminate redundancy in static information and focus on dynamic information for sub-pixel referenced compensation, in this paper, we propose a kernel-split strategy to activate available kernels for RTVSR. Unlike pruning schemes that discard internal parameters, we identify the high-value ones from feature channels to decrease the kernel dimension for a sparse processing pipeline. Specifically, a multi-channel selection unit is designed to split feature channels into low-value and high-value parts for static complementary and dynamic structural information respectively. In allusion to these allocated channels, a hierarchical architecture is adopted to filter the features step by step, reducing the kernel dimension and complexity of deep feature extraction. The

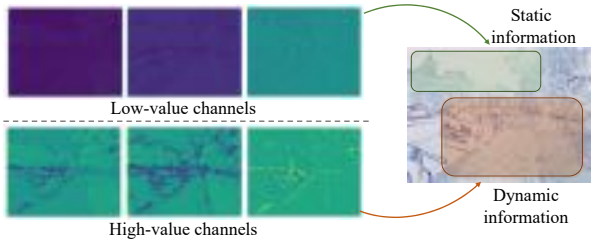


Figure 1: Visualization of features: low-value channels for the static information (green area) and high-value channels for the dynamic information (orange area).

re-parameterization mechanism is employed to merge the depth dimension of the convolutional kernel at each hierarchy, which increases the representation capacity and inference speed. Additionally, multiple flows are utilized to guide the deformable alignment. The training burden of deformable convolution is reduced via the residual offset strategy, and the representation of motion modelling is enhanced by continuous flow priors with affordable computational cost. Experimental results on different benchmarks verify that the proposed KSNet performs a real-time evaluation at 32.3 fps and outperforms other methods in terms of real-time reconstruction performance. Our contributions are summarized as follows:

- We propose a Kernel Split Network (KSNet) for RTVSR, including unidirectional and bidirectional recurrent paradigms, which can reconstruct the 720×1080 video with 31 ms and achieve superior results over other SOTA methods.
- We design a kernel-split strategy to discriminate the channels of high-value and low-value. The re-parameterized convolutions are further applied on the high-value channels. This strategy enables the exploration of high-value dynamic information and the representation of convolutional layers, along with dimension complexity reduction.
- We adopt multiple flows on the deformable alignment module to light the training burden of the deformable convolution and enhance the motion representation with affordable calculation costs. Multiple flow maps also expand the receptive field on the temporal dimension.

2 RELATED WORK

Existing video super-resolution methods can be categorized as plain video super-resolution methods which focus on the reconstruction quality, and real-time video super-resolution methods which focus on the runtime. In this section, we discuss these two categories and analyze the different strategies in detail.

2.1 Plain Video Super-resolution

Plain video super-resolution methods mostly employ complicated motion estimation and compensation strategies to explicitly explore the temporal correspondence to improve the reconstruction quality. In particular, Yan *et al.* [28] coupled the sliding window and unidirectional network to incorporate previous restored information and local information. Li *et al.* [12] introduced an iterative network

with the multi-scale flow estimation and complex reconstruction module for video super-resolution. Wang *et al.* [22] reconstructed the low-resolution optical flow result to improve the frame alignment performance for video super-resolution, and further proposed an optical flow reconstruction network (OFRNet) [23] to obtain the high-quality flow map for accurate alignment results. Chan *et al.* [3] further proposed a bidirectional framework to utilize the complementary motion information for video super-resolution. Since the accuracy of frame alignment depended on flow estimation, many methods explored implicit estimation strategies to align the consecutive frames. Wang *et al.* [24] presented a multi-scale deformable convolution using the pyramid structure to enhance the frame alignment. Ying *et al.* [32] further developed the 3D deformable convolution for the spatio-temporal information. Isobe *et al.* [8] employed 3D convolution blocks on decomposed frame groups for the inter-group alignment. In addition, some works utilized global temporal information via a non-local mechanism. Yi *et al.* [31] proposed to capture the spatio-temporal information among multi-frames. Liu *et al.* [15] proposed to learn the trajectory of the video contents using the self-attention mechanism. However, these methods relied on complicated modules and required complex calculations, which hindered the real-time evaluation.

2.2 Real-time Video Super-resolution

Existing real-time video super-resolution methods avoid using complicated calculation modules and deep networks to accelerate the inference speed. Specifically, Caballero *et al.* [20] proposed a novel sub-pixel convolutional layer to replace the interpolation algorithm for upsampling, and further introduced the lightweight motion estimation and sub-networks [2] to achieve the real-time inference. Dario *et al.* [6] designed a recurrent propagation mechanism in the latent space to incorporate the spatio-temporal information. Isobe *et al.* [9] proposed a recurrent identity mapping operation to capture the long-range information and accelerate the convergence. Yi *et al.* [30] just lowered the feature extractors in the omniscient network to achieve a real-time evaluation on the premise of promising reconstruction quality. Dario *et al.* [5] proposed an efficient alignment module using non-local attention for the temporal information. Zeng *et al.* [33] introduced a depthwise separable up-sampling module for real-time processing. Nevertheless, these methods discarded the temporal information to some extent and only achieved limited performance gains. In contrast, some methods utilized the pruning schemes on the deep networks to efficiently preserve temporal information. Zhang *et al.* [34] used a normalization layer on the scale parameter for sparse processing. Xia *et al.* [26] designed a pruning scheme to learn the sparse representation of VSR networks. We focus on the convolutional kernel dimensions to reduce the model redundancy, which is different from the sparsity-based pruning schemes and retains the temporal information.

3 PROPOSED METHOD

3.1 Overview

As shown in Figure 2, our network is constituted with a recurrent structure. Given a sequence of LR frames $\{x_1, x_2, ..., x_n, ..., x_N\}$ where $x_n \in \mathbb{R}^{C\times H\times W}$, the HR frames $\{y_1, y_2, ..., y_n, ..., y_N\}$ are reconstructed in a frame-to-frame pipeline. First, for an input frame x_n, h_n is the aligned feature warped from the previous frame x_{n-1}, and g_n is exploited for propagation. g_{n-1} is concatenated with h_n and x_n to obtain the coupled feature f for kernel split in the multi-channel selection unit, which can merge the kernel dimension and greatly reduce the model redundancy. The low-value channels are cached and reused for supplementary and the high-value channels are deeply explored for dynamic information using the re-parameterization strategy. Next, the propagation features $\{g_1, g_2, ..., g_n, ..., g_N\}$ are refined by the other recurrent branch with a similar process. Finally, a reconstruction module including residual blocks and a pixel shuffle layer [20] reconstructs HR frames.

3.2 Kernel-split Strategy

To address the issue of limited utilization of dynamic information and convolutional kernels, a Multi-channel Selection Unit (MSU) is devised to split the high-value and low-value channels to merge the kernel dimensions hierarchically.

Multi-channel Selection. Since the propagation feature g_{n-1} is concatenated with h_n and x_n to obtain the coupled feature f, some offsets still exist. These offsets represent the temporal content for inter-frame compensation and the dynamic information for temporal consistency. Thus, it is necessary to split the high-value and low-value channels from f to learn the dynamic information and reduce the model redundancy. In this approach, we use the weighted gradient map as a measure of temporal information and split channels according to the Euclidean norm. Specifically, we first calculate the gradient map of f to discriminate dynamic and static regions, which is formulated as:

$$G = \|\nabla f\|_2, \tag{1}$$

where G and ∇f denote the gradient value and gradient vector respectively. Then, the gradient map is masked with f, and weights are assigned to process different regions accordingly, which is shown in Figure 3. Next, we extract the original offset f_o from every two input frames, which represents the original temporal contents. The Euclidean distance $Euc(f_o, f)$ is employed to measure the channel correlations, formulated as:

$$Euc(f_o, f) = \|(\lambda_1 * G + (1-\lambda_1) * [G, f]) - f_o\|_2, \tag{2}$$

where $[\ ,\ ]$ denotes the mask operation and λ_1 is a learnable weight factor to control the trade-off between dynamic and static information, initialized as 0.5. Then, we select the top-K channels of f that are most correlated to the original offset f_o to exploit the dynamic information. The top 75% channels are marked as high-value channels and the remaining ones are marked as low-value channels, which will be verified in the ablation study section. The top-K selection strategy is formulated as follows.

$$top\text{-}k(f) = \begin{cases} f^{con} & \text{if the score of } Euc(f_o, f) \text{ is the top 75\%} \\ f^{dis} & \text{others} \end{cases} \tag{3}$$

Hierarchical Kernel-split. After the multi-channel selection, the coupled feature f is split into sub-features $\{f^{dis}, f^{con}\}$, where f^{dis} represents the low-value channels and f^{con} represents the high-value channels. For hierarchical processing with a three-level stage, the high-value channels are split following Eq. (3) at each hierarchy, as shown in Figure 4(a). Take the first hierarchy for example, the channel number of f_1^{dis} is split as a quarter of f, and

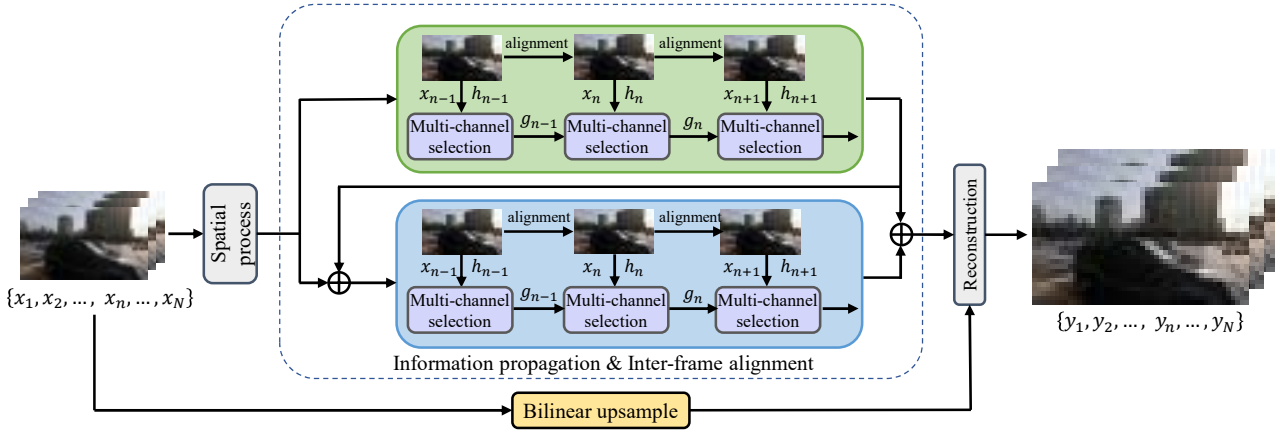
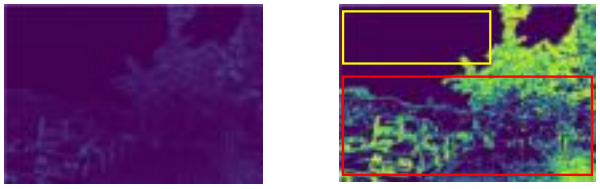


Figure 2: Overview of the proposed KSNet, which is a recurrent-based framework with a frame-to-frame pipeline. The input frames are propagated in the first recurrent branch (green area) and then refined in the other recurrent branch (blue area).

the retained channels are allocated to f_1^{con}. A Shallow Convolution Buffer (SCB) is utilized on f_1^{dis} to cache and reuse the low-value channels as well as to decrease the kernel dimensions. The high-value channels of f_1^{con} are deeply exploited to obtain f_2^{con} for further processing. The kernel dimension is reduced along with the split channels. Finally, the cached low-value features and high-value features are exploited by the Enhanced Spatial Attention (ESA) [16] to obtain the propagation feature g. The procedure of MSU is formulated as:

$$\begin{aligned} f_{dc} &= cat(f_1^{dis}, f_2^{dis}, f_3^{dis}, f_3^{con}), \\ g &= ESA(cat(f, f_{dc})), \end{aligned} \tag{4}$$

where f_1^{dis}, f_2^{dis} and f_3^{dis} denote the low-value channels at different hierarchy respectively, and f_3^{con} denotes the high-value feature at the last hierarchy. In this procedure, f_3^{con} is filtered step by step to select the dynamic information for propagation, where f_1^{dis}, f_2^{dis} and f_3^{dis} are cached in the buffer to provide the static complementary information. This decreases the redundant kernels for feature exploitation and propagation.



(a) Gradient map (b) Weighted gradient map

Figure 3: (a) is the gradient map G. (b) is the weighted map combined with G and f. The highlighted area in the red rectangle represents the temporal contents with high weights. The other area in the yellow rectangle deserves low weights.

Moreover, ESA is utilized to enhance the representation of the deep features on the spatial dimension. While MSU splits the high-value and the low-value channels to explore dynamic information on the channel dimension, the correspondence on the spatial dimension also deserves attention. Since the shallow and the deep features require to be focused on spatial contents of key importance, they can be exploited by the spatial attention mechanism. Specifically, ESA works at the end of MSU to aggregate the shallow and deep features which are forced to focus on the regions of interest.

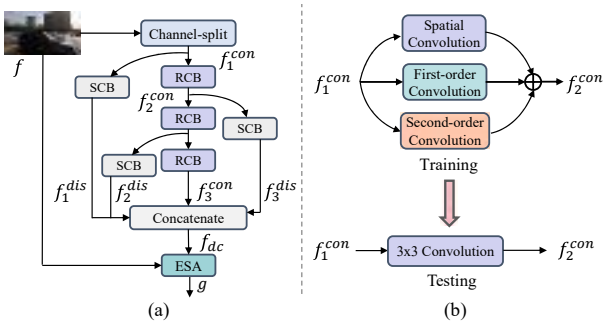


Figure 4: Multi-channel selection unit. (a) is the hierarchical split structure. (b) is the re-parameterized convolution block which is employed in MSU to reduce the computational cost.

Kernel Re-parameterization. Solely using a 3×3 convolutional layer for deep feature exploration leads to a weak representation. Hence, we employ the Re-parameterized Convolution Block (RCB) on the high-value channels for deep feature extraction, as shown in Figure 4(b). Parallel convolutions are adopted to replace the single vanilla convolution. These convolutions include a plain convolutional kernel to extract the spatial information, and first-order and second-order edge convolutional kernels to extract the detailed information. The first-order and second-order convolutional kernels are high-order versions since they calculate spatial derivatives for edge details. Specifically, we incorporate extraction of derivatives into the spatial convolutional kernel to get dynamic structures, where the Sobel filter and the Laplacian filter are employed to calculate the first-order and the second-order derivatives, respectively.

These convolutional kernels are merged as a single convolutional kernel at the inference stage, which can greatly reduce the calculation complexity.

We take the sequential convolutional kernels for example, which are employed to enlarge the representation capacity at the training stage and re-parameterized as the single convolutional layer at the inference stage for efficiency. The consecutive convolutions are formulated as:

$$\sigma_2(\sigma_1(k)) = W_2 * (W_1 * k + B_1) + B_2, \tag{5}$$

where W_1 and W_2 denote the weights of the convolutional kernel. B_1 and B_2 denote the bias of the convolution. The σ_1 convolution (W_1, B_1) and σ_2 convolution (W_2, B_2) are merged into the re-parameterized version, formulated as:

$$\begin{aligned} W_{\text{ps}} &= perm(W_1) * W_2, \\ B_{\text{ps}} &= W_2 * rep(B_1) + B_2, \end{aligned} \tag{6}$$

where $perm$ and rep denote the dimension exchange and broadcasting operation to maintain the tenor size. The transformation of the re-parameterized kernel is formulated as:

$$\begin{aligned} W_{\text{pr}}^{:,:,i+\lfloor \frac{H-h}{2} \rfloor, j+\lfloor \frac{W-w}{2} \rfloor} &= W_{\text{ps}}^{:,:,i,j}, \\ B_{\text{pr}} &= B_{\text{ps}}, \end{aligned} \tag{7}$$

where i and j denote the element position in the convolutional kernels. W_{pr} and B_{pr} denote the target convolution with the kernel $H \times W$, transformed from the kernel $h \times w$. The final weights W_{re} and bias B_{re} are re-parameterized as:

$$\begin{aligned} W_{re} &= W_p + W_{one} + W_{sec}, \\ B_{re} &= B_p + B_{one} + B_{sec}, \end{aligned} \tag{8}$$

where W_{re} and B_{re} denote the weights and bias of the re-parameterized convolutional kernels. W_{one}, W_{sec} and B_{one}, B_{sec} denote the weights and bias from the first- and second-order convolutional kernels respectively, which are transformed following Eq. (6) and Eq.(7).

3.3 Deformable Alignment with Multiple Flow

To meet the real-time requirement, some efficient networks neglect the inter-frame information to decrease the complexity, leading to blurs and limited performance gains. It is essential to design an efficient alignment method with affordable computational cost. Thus, a Multiple Flow Deformable Alignment (MFDA) module is proposed to align the motion area, as shown in Figure 5.

For features f_n, f_{n-1} and f_{n-2} that are extracted from the corresponding frames x_n, x_{n-1} and x_{n-2}, flow maps between the successive frames $s_{n\to n-1}$ and $s_{n-1\to n-2}$ are estimated as:

$$\begin{aligned} s_{n\to n-1} &= \text{ME}(f_n, f_{n-1}), \\ s_{n-1\to n-2} &= \text{ME}(f_{n-1}, f_{n-2}), \end{aligned} \tag{9}$$

where ME denotes the optical flow network. The optical flow $s_{n\to n-1}$ indicates the flow offsets from f_{n-1} to f_n. In other words, $s_{n\to n-1}$ represents the pixel shift of f_n compared with f_{n-1}. The multi-reference flow $\bar{s}_n$ is obtained by the sum of $s_{n\to n-1}$ and the warped $s_{n-1\to n-2}$.

$$\bar{s}_n = s_{n\to n-1} + \text{MC}(s_{n-1\to n-2}, s_{n\to n-1}), \tag{10}$$

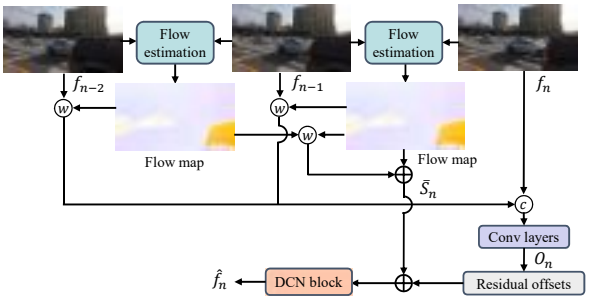


Figure 5: Architecture of the proposed deformable alignment with multiple flows. Flow maps are calculated from different inter-frames to extend the temporal receptive field.

where MC denotes the motion compensation. Meanwhile, f_{n-1} and f_{n-2} are warped with $s_{n\to n-1}$ and $s_{n-1\to n-2}$ respectively to obtain the multi-reference feature $\overline{f}_n$.

$$\overline{f}_n = \text{MC}(f_{n-2}, s_{n-1\to n-2}) + \text{MC}(f_{n-1}, s_{n\to n-1}), \tag{11}$$

Finally, the residual offset O_n is computed from $\overline{f}_n$ and f_n. The aligned feature $\hat{f}_n$ from f_{n-1} to f_n is computed from the multi-reference flow $\bar{s}_n$ and offset O_n using the deformable convolution.

$$\begin{aligned} O_n &= C(cat(\overline{f}_n, f_n)), \\ \hat{f}_n &= D(O_n + \bar{s}_n), \end{aligned} \tag{12}$$

where C denotes the convolutional layer and D denotes the DCN layer. We utilize the lightweight optical flow network SpyNet [19] for motion estimation and shallow convolutional layers for DCN offsets since it can conduct efficient flow estimation with affordable time consumption.

4 EXPERIMENT

4.1 Datasets and Implementation Details

Comprehensive experiments are conducted on different datasets. REDS [18] and Vimeo90K [27] are used for training. For REDS [18], we adopt REDS4 [18] as test dataset following the previous work [24]. For Vimeo90K [27], we take Vid4 [14] and Vimeo90K-T [27] as test datasets. For a fair comparison, all of the models are tested using one NVIDIA RTX 3080.

At the training stage, the cosine annealing scheme [17] is adopted to adjust the learning rate. The initial learning rate of the main network is set to 1×10^{-4} on REDS [18] and 2×10^{-4} on Vimeo90K [27], while the learning rate of the fine-tuned flow network is doubled. The total iteration number is 600K. The weights of the flow estimator are fixed during the first 5,000 iterations. We use ADAM optimizer [10] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The patch size in training is set to 64×64 and the channel number of features maps is set to 64. We adopt 5 residual blocks for spatial processing and 1 residual block for reconstruction. The Charbonnier loss [11] is used to constrain a stable network, formulated as:

$$L = \frac{1}{N}\sum_{n=1}^{N}\sqrt{(y_n - Y_n)^2 + \varepsilon^2}, \tag{13}$$

where Y_n is the n-th ground truth frame and ε is set to 10^{-3}.

Table 1: Quantitative comparison with different methods. The best and second-best results are marked with bold and underline respectively. FLOPs (G), fps (1/s), and runtime (ms) are computed on an LR size of 180×320 using one NVIDIA RTX 3080.

Methods	Params(M)	FLOPs(G)	fps(1/s)	Run(ms)	Test datasets		
					Vid4	REDS4	Vimeo90K-T
Bicubic	-	-	-	-	23.78/0.6347	26.14/0.7292	31.30/0.8687
VESPCN [2]	-	-	28.6	35	25.35/0.7557	-	-
RLSP [6]	4.2	503.7	21.7	46	27.05/0.8139	-	36.49/0.9403
RRN [9]	3.4	387.5	22.2	45	27.09/0.8185	-	-
DAP-128 [5]	-	330.0	26.3	38	-	30.59/0.8703	37.29/0.9476
RSDN [7]	6.2	713.2	10.6	94	**27.22/0.8249**	-	37.23/0.9471
EDVR-M [24]	3.3	925.7	8.6	116	27.10/0.8186	30.53/0.8699	37.33/0.9484
KSNet-uni (Ours)	**3.0**	**296.9**	**32.3**	**31**	27.14/0.8208	**30.69/0.8724**	**37.34/0.9490**
ASSL-bi [34]	2.7	210.2	32.3	31	27.03/0.8163	30.72/0.8783	36.71/0.9410
RSCL-bi [26]	**2.7**	**210.2**	32.3	31	27.16/0.8213	30.99/0.8831	36.83/0.9421
BasicVSR [3]	6.3	330.0	15.9	63	**27.24/0.8251**	**31.42/0.8909**	37.53/0.9450
KSNet-bi (Ours)	3.0	296.9	**32.3**	**31**	27.22/0.8245	31.14/0.8862	**37.54/0.9503**

4.2 Comparisons with State-of-the-Art

We train and test KSNet with two setups: KSNet-uni and KSNet-bi for unidirectional and bidirectional propagation. Parameter numbers (Params), FLOPs, frames per second (fps), and runtime (Run) are used to evaluate the efficiency, while PSNR and SSIM are employed to measure the reconstruction quality. We compare KSNet-uni with RLSP [6], RRN [9], RSDN [7], etc., and KSNet-bi with BasicVSR [3] and its pruned versions [26, 34].

The results are summarized in Table 1. As a unidirectional type, KSNet-uni performs a real-time evaluation with 31 ms and 32.3 fps, which is the only method that exceeds 30 fps. Moreover, KSNet-uni almost surpasses other methods on all metrics. KSNet-uni gets 0.1 dB gains than the real-time method DAP [5]. KSNet-uni further surpasses RSDN [7] 0.11 dB on Vimeo90K-T [27] and gets a comparable result on Vid4 [14], with a reduction in both runtime and FLOPs over × 3. As a bidirectional type, KSNet-bi achieves the same inference speed and superior performance compared with pruning-based methods. Specifically, KSNet-bi outperforms RSCL-bi [26] up to 0.7 dB on Vimeo90K-T [27] with the same runtime. Besides, compared to the plain method BasicVSR [3], KSNet-bi gets 0.01 dB gains on Vimeo90K-T [27] and comparable results on Vid4 [14] with half of the runtime. These results show that KSNet performs a real-time inference with competitive reconstruction quality.

(a) low-value feature (b) high-value feature

Figure 6: Visualization of the high-value and low-value features. The red rectangle in (b) contains precise motions, while the red rectangle in (a) only contains mixed offsets.

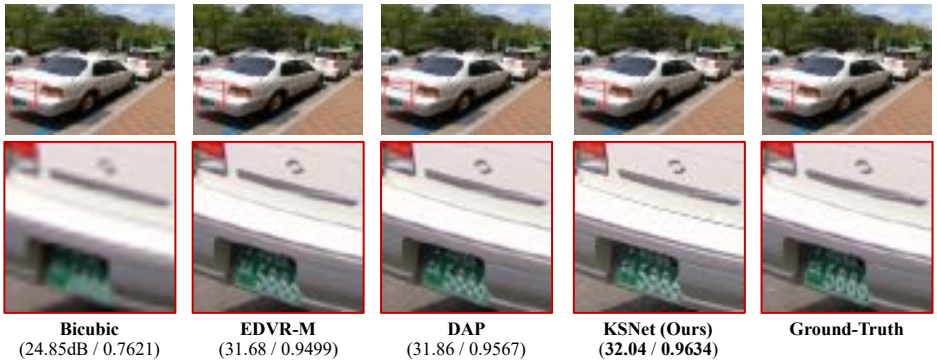
Moreover, we visualize the high-value and low-value features to evaluate the dynamic information in Figure 6. It's seen that motion offsets in the low-value feature are blurred and mixed. On the contrary, the high-value feature contains clear offsets and concrete structures to learn the temporal information. The consecutive detailed offsets also represent the motion trajectory for temporal consistency. Therefore, the high-value feature that consists of precise dynamic information deserves more attention and exploration. As shown in Figure 7, we also provide some visual samples for qualitative evaluation. Bicubic interpolation, the plain method EDVR-M [24] and the real-time method DAP [5] are taken into comparison. KSNet produces high-quality frames with clear structure and edge details, while other methods generate artifacts at different levels. These samples also verify the superiority of KSNet.

5 ABLATION STUDY

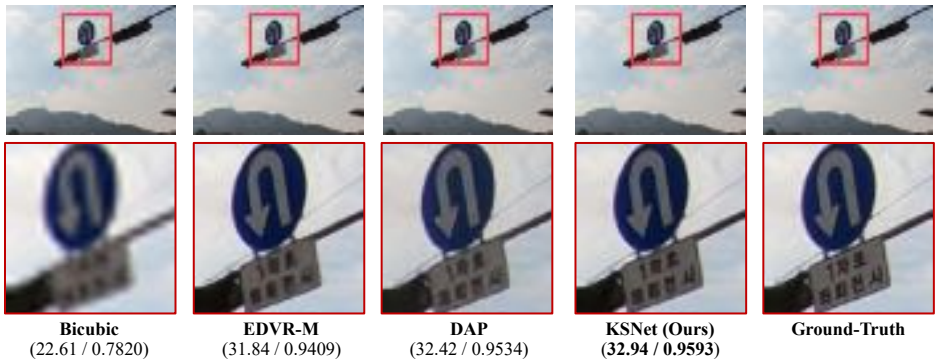
5.1 Kernel Split Ratio

The MSU splits feature channels into high-value and low-value parts and allocates different convolutional kernels for efficient processing. It is inappropriate to determine the optimal split ratio empirically. Therefore, we conduct ablation experiments to confirm the optimal ratio in KSNet, which is shown in Figure 8.

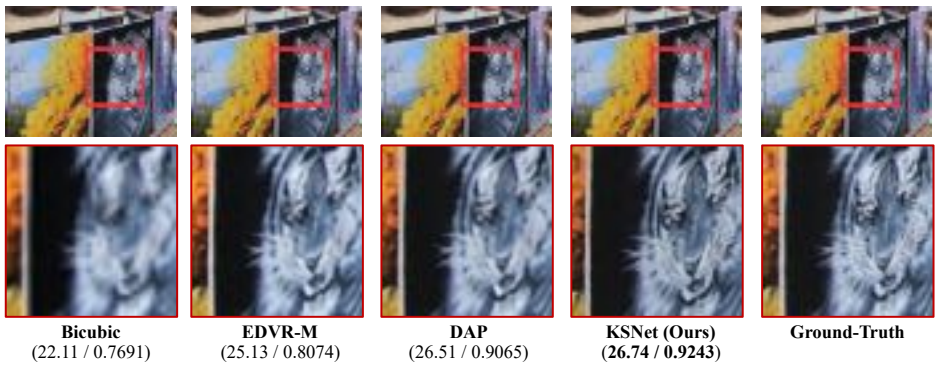
It is seen that increasing the split ratio on the high-value channel improves reconstruction performance. Specifically, while the split ratio is below 50%, KSNet gets limited performance gains with less than 3.0 M parameters, indicating that the constrained ratio hinders the performance gains. The 100% ratio also obtains limited performance gains and needs extra 0.3 M parameters compared to the 75% ratio. It indicates that the dynamic information of high-value channels tends to be saturated in the 75% feature channels. Increasing the split ratio beyond this point doesn't significantly improve the restoration performance but rather adds redundant parameters. These results show that the 75% split ratio strikes the trade-off between restoration performance and model complexity. Moreover, we take visual samples to verify the reconstruction quality, shown in Figure 9. It's seen that the 100% setup obtains the best visual quality compared to the others. 25% and 50% setups generate blurry artifacts in brickwork textures. 75% setup also generates visual-pleasing results and strikes the trade-off between reconstruction quality and parameter numbers.



(a) **Visual samples on REDS4 (*000 clip*)**



(b) **Visual samples on REDS4 (*010 clip*)**



(c) **Visual samples on REDS4 (*020 clip*)**

Figure 7: Visual comparisons with SOTA methods on REDS dataset.

5.2 Re-parameterized Convolutional Kernel

To evaluate the effectiveness of the re-parameterization strategy, we design a plain topology with two setups: KSNet-uni-plain and KSNet-bi-plain, denoted as base networks, which only contains a 3 × 3 convolutional layer to substitute the re-parameterized convolutional kernels. We take parameter numbers (Params) and FLOPs at the training and testing stages as metrics for model analysis. Runtime (Run), frames per second (fps) and PSNR are used to evaluate the efficiency and reconstruction performance.

The numerical comparisons are presented in Table 2. It can be observed that the plain models, including KSNet-uni-plain and KSNet-bi-plain, have different parameter numbers and FLOPs at the training stages compared with the re-parameterized model. This discrepancy arises due to the fact that plain models only contain a single convolutional kernel in RCB, while the re-parameterized models contain parallel convolutional kernels. These kernels are re-parameterized as a single kernel during testing. In other words, the structure of plain and re-parameterized models remains the same

Table 2: Ablation study on re-parameterization strategy.

Model	Params-training(M)	Params-testing(M)	FLOPs-training(T)	FLOPs-testing(T)	Run(ms)	fps(1/s)	PSNR(dB)
KSNet-uni-plain	3.0	3.0	296.9	296.9	31	32.3	30.50
KSNet-bi-plain	3.0	3.0	296.9	296.9	31	32.3	30.94
KSNet-uni (Ours)	3.4	3.0	320.9	296.9	31	32.3	30.69
KSNet-bi (Ours)	3.4	3.0	320.9	296.9	31	32.3	31.14

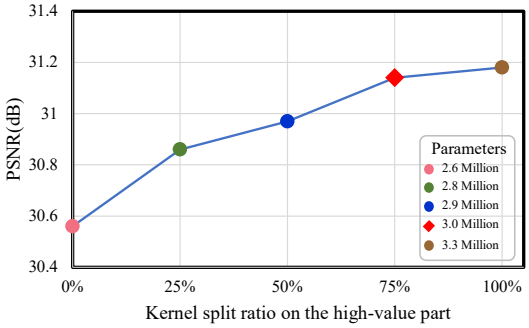


Figure 8: Comparisons on the reconstruction performance and kernel split ratio. Color dots represent the parameter number of different configurations.

Figure 9: Subjective comparisons on the visual quality of different split ratios.

at the inference stage, leading to equivalent parameters, FLOPs, and runtime. Moreover, compared to plain models, KSNet-uni and KSNet-bi achieve respective gains of 0.19 dB and 0.2 dB with only 17.2% additional parameters and 8% additional FLOPs in training. These results demonstrate that the re-parameterized convolutional kernel promotes the representation capacity of the network without imposing extra inference burdens.

5.3 Multiple Flows for Deformable Alignment

To assess the capacity of flow priors in deformable alignment, several ablation experiments are conducted on REDS[18] dataset. Model A to Model D denote different networks, including the single flow setup $SF_{n\to n-1}$ from x_{n-1} to x_n and $SF_{n\to n-2}$ from x_{n-2} to x_n, and the multi-flow setup MF. We use runtime (Run) and frames per second (fps) to evaluate the efficiency of different setups.

The results are listed in Table 3. Compared to Model A which solely utilizes the deformable alignment without flow priors, adding single flows $SF_{n\to n-2}$ and $SF_{n\to n-1}$ leads to performance gains of 0.28 dB and 0.45 dB for Model B and Model C respectively. These results indicate that the single flow prior is effective in guiding the deformable alignment. Furthermore, Model C outperforms Model B by 0.17 dB, which demonstrates that the accuracy of the single flow is dependent on the frame interval and large intervals contain complex offsets for flow estimation. The sub-pixel compensation is more precise in a shorter frame interval for the single flow prior.

In addition, Model D uses multiple flows (MF), including coherent $SF_{n\to n-2}$ and $SF_{n\to n-1}$. It's seen that Model D achieves a significant gain of 0.81 dB compared to Model A. Compared to Model B and Model C, Model D achieves respective improvements of 0.53 dB and 0.36 dB with only an additional 1 ms runtime requirement. These results demonstrate that incorporating more inter-frame information enhances the deformable alignment with minimal extra computational cost.

Table 3: Ablation study on different flow settings.

Mechanisms	$SF_{n\to n-2}$	$SF_{n\to n-1}$	MF	Run (ms)	fps (1/s)	PSNR (dB)
Model A	✘	✘	✘	26	38.5	30.33
Model B	✔	✘	✘	30	33.3	30.61
Model C	✘	✔	✘	30	33.3	30.78
Model D	✘	✘	✔	31	32.3	31.14

6 CONCLUSION

In this paper, we propose a Kernel Split Network (KSNet) to activate available kernels for RTVSR. A multi-channel selection unit is designed to split the low-value and high-value features to reduce redundancy. Re-parameterized convolutions are applied to the high-value channels, which can merge the kernel depth dimension and enhance the representation capacity. An efficient deformable alignment module with multiple flows is further presented to leverage the temporal information with affordable extra complexity. Experimental results verify the superior performance of KSNet over other SOTA methods in terms of runtime and reconstruction quality.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (Grant No. 2022YFE0129200) and the National Natural Science Foundation of China (Grant No.61972028, 61902022, and 62120106009).

REFERENCES

[1] Bahetiyaer Bare, Bo Yan, Chenxi Ma, and Ke Li. 2019. Real-time video super-resolution via motion convolution kernel estimation. *Neurocomputing* 367 (2019), 236–245.

[2] Jose Caballero, Christian Ledig, Andrew P. Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *CVPR*. 2848–2857.

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

[4] Kelvin CK Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. 2022. BasicVSR++: Improving video super-resolution with enhanced propagation and alignment. In *CVPR*. 5972–5981.

[5] Dario Fuoli, Martin Danelljan, Radu Timofte, and Luc Van Gool. 2023. Fast online video super-resolution with deformable attention pyramid. In *ACCV*. 1735–1744.

[6] Dario Fuoli, Shuhang Gu, and Radu Timofte. 2019. Efficient video super-resolution through recurrent latent space propagation. In *ICCVW*. 3476–3485.

[7] Takashi Isobe, Xu Jia, Shuhang Gu, Songjiang Li, Shengjin Wang, and Qi Tian. 2020. Video super-resolution with recurrent structure-detail network. In *ECCV*. 645–660.

[8] Takashi Isobe, Songjiang Li, Xu Jia, Shanxin Yuan, Gregory Slabaugh, Chunjing Xu, Ya-Li Li, Shengjin Wang, and Qi Tian. 2020. Video super-resolution with temporal group attention. In *CVPR*. 8008–8017.

[9] Takashi Isobe, Fang Zhu, Xu Jia, and Shengjin Wang. 2020. Revisiting temporal modeling for video super-resolution. *arXiv preprint arXiv:2008.05765* (2020).

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

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

[12] Feng Li, Huihui Bai, and Yao Zhao. 2020. Learning a deep dual attention network for video super-resolution. *IEEE TIP* 29 (2020), 4474–4488.

[13] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2017. Pruning filters for efficient convnets. In *ICLR*.

[14] Ce Liu and Deqing Sun. 2013. On Bayesian adaptive video super resolution. *IEEE TPAMI* 36, 2 (2013), 346–360.

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

[16] Jie Liu, Wenjie Zhang, Yuting Tang, Jie Tang, and Gangshan Wu. 2020. Residual feature aggregation network for image super-resolution. In *CVPR*. 2359–2368.

[17] Ilya Loshchilov and Frank Hutter. 2016. SGDR: stochastic gradient descent with warm restarts. *arXiv Preprint arXiv:1608.03983* (2016).

[18] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. 2019. NTIRE 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPRW*.

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

[20] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*. 1874–1883.

[21] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. 2017. Detail-revealing deep video super-resolution. In *ICCV*. 4472–4480.

[22] Longguang Wang, Yulan Guo, Zaiping Lin, Xinpu Deng, and Wei An. 2019. Learning for video super-resolution through HR optical flow estimation. In *ACCV*. 514–529.

[23] Longguang Wang, Yulan Guo, Li Liu, Zaiping Lin, Xinpu Deng, and Wei An. 2020. Deep video super-resolution using HR optical flow estimation. *IEEE TIP* 29 (2020), 4323–4336.

[24] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. 2019. EDVR: Video restoration with enhanced deformable convolutional networks. In *CVPRW*.

[25] Zhongyuan Wang, Peng Yi, Kui Jiang, Junjun Jiang, Zhen Han, Tao Lu, and Jiayi Ma. 2018. Multi-memory convolutional neural network for video super-resolution. *IEEE TIP* 28, 5 (2018), 2530–2544.

[26] Bin Xia, Jingwen He, Yulun Zhang, Yucheng Hang, Wenming Yang, and Luc Van Gool. 2022. Residual sparsity connection learning for efficient video super-resolution. *arXiv preprint arXiv:2206.07687* (2022).

[27] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. 2019. Video enhancement with task-oriented flow. *IJCV* 127, 8 (2019), 1106–1125.

[28] Bo Yan, Chuming Lin, and Weimin Tan. 2019. Frame and feature-context video super-resolution. In *AAAI*, Vol. 33. 5597–5604.

[29] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, Tao Lu, and Jiayi Ma. 2020. A progressive fusion generative adversarial network for realistic and consistent video super-resolution. *IEEE TPAMI* 44, 5 (2020), 2264–2280.

[30] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, Tao Lu, Xin Tian, and Jiayi Ma. 2021. Omniscient video super-resolution. In *ICCV*. 4429–4438.

[31] Peng Yi, Zhongyuan Wang, Kui Jiang, Junjun Jiang, and Jiayi Ma. 2019. Progressive fusion video super-resolution network via exploiting non-local spatio-temporal correlations. In *ICCV*. 3106–3115.

[32] Xinyi Ying, Longguang Wang, Yingqian Wang, Weidong Sheng, Wei An, and Yulan Guo. 2020. Deformable 3D convolution for video super-resolution. *IEEE SPL* 27 (2020), 1500–1504.

[33] Yubin Zeng, Zhijiao Xiao, Kwok-Wai Hung, and Simon Lui. 2021. Real-time video super resolution network using recurrent multi-branch dilated convolutions. *Signal Processing: Image Communication* 93 (2021), 116167.

[34] Yulun Zhang, Huan Wang, Can Qin, and Yun Fu. 2021. Aligned structured sparsity learning for efficient image super-resolution. In *NeurIPS*, Vol. 34. 2695–2706.