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

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
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. Dairo et al. [6] propose a lightweight convolutional kernel to reduce the time complexity. Isobe et al. [9] present a recurrent identity mapping operation to preserve long-range information. Some methods design lightweight alignment strategies to utilize temporal information. Bare et al. [1] propose a lightweight convolutional kernel for alignment and introduce 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] propose 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 complement 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 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 $720 \times 1080$ 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 (OFRNNet) [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-frame 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 \(\{x_1, x_2, ..., x_N\}\) where \(x_n \in R^{C \times H \times W}\), the HR frames \(\{y_1, y_2, ..., y_N\}\) 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 \(\{g_1, g_2, ..., g_N\}\) 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:

\[
G = \| \nabla f \|_2,
\]

where \(G\) and \(\nabla f\) 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 \(f_\text{ori}\) from every two input frames, which represents the original temporal contents. The Euclidean distance \(Euc(f_\text{ori}, f)\) is employed to measure the channel correlations, formulated as:

\[
Euc(f_\text{ori}, f) = \| (\lambda_1 G + (1 - \lambda_1) \ast [G, f]) - f_\text{ori} \|_2.
\]

where \([\cdot, \cdot]\) denotes the mask operation and \(\lambda_1\) 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 \(f_\text{ori}\) 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.

\[
top-k(f) = \begin{cases} f_{\text{con}} & \text{if the score of } Euc(f_\text{ori}, f) \text{ is the top 75\%} \\ f_{\text{dis}} & \text{others} \end{cases}
\]

**Hierarchical Kernel-split.** After the multi-channel selection, the coupled feature \(f\) is split into sub-features \(\{f_{\text{dis}}, f_{\text{con}}\}\), where \(f_{\text{dis}}\) represents the low-value channels and \(f_{\text{con}}\) 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 \(f_{\text{dis}}^{1}\) 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 $f_{1con}$. A Shallow Convolution Buffer (SCB) is utilized on $f_{1dis}$ to cache and reuse the low-value channels as well as to decrease the channel dimensions. The high-value channels of $f_{2con}$ are deeply exploited to obtain $f_{2con}$ 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:

$$
\begin{align*}
    f_{dc} &= cat(f_{1dis}, f_{2dis}, f_{3dis}, f_{3con}), \\
    g &= ESA(cat(f, f_{dc})),
\end{align*}
$$

where $f_{1dis}$, $f_{2dis}$ and $f_{3dis}$ denote the low-value channels at different hierarchy respectively, and $f_{3con}$ denotes the high-value feature at the last hierarchy. In this procedure, $f_{3con}$ is filtered step by step to select the dynamic information for propagation, where $f_{1dis}$, $f_{2dis}$ and $f_{3dis}$ are cached in the buffer to provide the static complementary information. This decreases the redundant kernels for feature exploitation and propagation.

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.

Kernel Re-parameterization. Solely using a 3x3 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 convolution 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 calcu-
lation 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:

\[ \sigma_2(\sigma_1(k)) = W_2 \ast (W_1 \ast k + B_1) + B_2, \]

(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 \( \sigma_1 \) con-
volution (\( W_1, B_1 \)) and \( \sigma_2 \) convolution (\( W_2, B_2 \)) are merged into the
re-parameterized version, formulated as:

\[
\begin{align*}
W_{ps} &= \text{perm}(W_1) + W_2, \\
B_{ps} &= W_2 \ast \text{rep}(B_1) + B_2,
\end{align*}
\]

(6)

where \( \text{perm} \) and \( \text{rep} \) denote the dimension exchange and broad-
casting operation to maintain the tenor size. The transformation of
the re-parameterized kernel is formulated as:

\[
\begin{align*}
W_{pr}^{i,j} &= \frac{W_{ps}^{i,j} \cdot \frac{\text{MC}}{2}}{\frac{\text{MC}}{2} + \frac{\text{MC}}{2}}, \\
B_{pr} &= B_{ps},
\end{align*}
\]

(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:

\[
\begin{align*}
W_{re} &= W_p + W_{one} + W_{sec}, \\
B_{re} &= B_p + B_{one} + B_{sec},
\end{align*}
\]

(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 re-
spectively, 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 corre-
sponding frames \( x_n, x_{n-1} \) and \( x_{n-2} \), flow maps between the success-
ive frames \( s_{n-1}, s_{n-2} \) and \( s_{n-1-\to n-2} \) are estimated as:

\[
\begin{align*}
s_{n-1} &= \text{ME}(f_n, f_{n-1}), \\
s_{n-1-\to n-2} &= \text{ME}(f_{n-1}, f_{n-2}),
\end{align*}
\]

(9)

where \( \text{ME} \) 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 \( \bar{t}_{n} \) is obtained by the sum of \( s_{n-\to n-1} \) and the warped
\( s_{n-1-\to n-2} \):

\[
\bar{t}_{n} = s_{n-\to n-1} + \text{MC}(s_{n-1-\to n-2}, s_{n-\to n-1}),
\]

(10)

where \( \text{MC} \) 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 \( \bar{f}_{n} \):

\[
\bar{f}_{n} = \text{MC}(f_{n-2}, s_{n-1-\to n-2}) + \text{MC}(f_{n-1}, s_{n-\to n-1}).
\]

(11)

Finally, the residual offset \( O_{n} \) is computed from \( \bar{f}_{n} \) and \( f_{n} \). The
aligned feature \( \hat{f}_{n} \) from \( f_{n-1} \) to \( f_{n} \) is computed from the multi-
reference flow \( \bar{t}_{n} \) and offset \( O_{n} \) using the deformable convolution.

\[
\hat{f}_{n} = \text{D}(O_{n} + \bar{t}_{n}),
\]

(12)

where \( \text{C} \) denotes the convolutional layer and \( \text{D} \) 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 net-
work is set to \( 1 \times 10^{-3} \) on REDS [18] and \( 2 \times 10^{-4} \) 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 esti-
mator 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 \( 64 \times 64 \) 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:

\[
L = \frac{1}{N} \sum_{n=1}^{N} \sqrt{(Y_n - \hat{Y}_n)^2 + \epsilon^2},
\]

(13)

where \( Y_n \) is the \( n \)-th ground truth frame and \( \epsilon \) 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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Params(M)</th>
<th>FLOPs(G)</th>
<th>fps(1/s)</th>
<th>Run(ms)</th>
<th>Vid4</th>
<th>REDS4</th>
<th>Vimeo90K-T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicubic</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>27.38/0.6347</td>
<td>26.14/0.7292</td>
<td>31.30/0.8687</td>
</tr>
<tr>
<td>VESPCN [2]</td>
<td>-</td>
<td>-</td>
<td>28.6</td>
<td>35</td>
<td>25.35/0.7557</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RLSP [6]</td>
<td>4.2</td>
<td>503.7</td>
<td>21.7</td>
<td>46</td>
<td>27.05/0.8139</td>
<td>-</td>
<td>36.49/0.9403</td>
</tr>
<tr>
<td>RRN [9]</td>
<td>3.4</td>
<td>387.5</td>
<td>22.2</td>
<td>45</td>
<td>27.09/0.8185</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DAP-128 [5]</td>
<td>-</td>
<td>330.0</td>
<td>26.3</td>
<td>38</td>
<td>-</td>
<td>30.59/0.8703</td>
<td>37.29/0.9476</td>
</tr>
<tr>
<td>RSDN [7]</td>
<td>6.2</td>
<td>713.2</td>
<td>10.6</td>
<td>94</td>
<td>27.22/0.8249</td>
<td>-</td>
<td>37.23/0.9471</td>
</tr>
<tr>
<td>EDVR-M [24]</td>
<td>3.3</td>
<td>925.7</td>
<td>8.6</td>
<td>116</td>
<td>27.10/0.8186</td>
<td>30.53/0.8699</td>
<td>37.33/0.9484</td>
</tr>
<tr>
<td>KSNet-uni (Ours)</td>
<td>3.0</td>
<td>296.9</td>
<td>32.3</td>
<td>31</td>
<td>27.14/0.8208</td>
<td>30.69/0.8724</td>
<td>37.34/0.9490</td>
</tr>
<tr>
<td>ASSL-bi [34]</td>
<td>2.7</td>
<td>210.2</td>
<td>32.3</td>
<td>31</td>
<td>27.03/0.8163</td>
<td>30.72/0.8783</td>
<td>36.71/0.9410</td>
</tr>
<tr>
<td>RSCL-bi [26]</td>
<td>2.7</td>
<td>210.2</td>
<td>32.3</td>
<td>31</td>
<td>27.16/0.8213</td>
<td>30.99/0.8831</td>
<td>36.83/0.9421</td>
</tr>
<tr>
<td>BasicVSR [3]</td>
<td>6.3</td>
<td>330.0</td>
<td>15.9</td>
<td>63</td>
<td>27.24/0.8251</td>
<td>31.42/0.8909</td>
<td>37.53/0.9450</td>
</tr>
<tr>
<td>KSNet-bi (Ours)</td>
<td>3.0</td>
<td>296.9</td>
<td>32.3</td>
<td>31</td>
<td>27.22/0.8245</td>
<td>31.14/0.8862</td>
<td>37.54/0.9503</td>
</tr>
</tbody>
</table>

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.

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 reconstruction 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.
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 x 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.

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

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

Table 3: Ablation study on different flow settings.

<table>
<thead>
<tr>
<th>Mechanisms</th>
<th>$SF_{n \rightarrow n-2}$</th>
<th>$SF_{n \rightarrow n-1}$</th>
<th>$MF$</th>
<th>Run (ms)</th>
<th>fps (1/s)</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model A</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>26</td>
<td>38.5</td>
<td>30.33</td>
</tr>
<tr>
<td>Model B</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>30</td>
<td>33.3</td>
<td>30.61</td>
</tr>
<tr>
<td>Model C</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>30</td>
<td>33.3</td>
<td>30.78</td>
</tr>
<tr>
<td>Model D</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>31</td>
<td>32.3</td>
<td>31.14</td>
</tr>
</tbody>
</table>

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


