Rethinking Alignment in Video Super-Resolution Transformers

Shuwei Shi[1,2,*], **Jinjin Gu**[3,4,*], **Liangbin Xie**[2,5,6], **Xintao Wang**[6], **Yujiu Yang**[1], **Chao Dong**[2,3,†]
[1] Shenzhen International Graduate School, Tsinghua University
[2] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[3] Shanghai AI Laboratory [4] The University of Sydney
[5] University of Chinese Academy of Sciences [6] ARC Lab, Tencent PCG
jinjin.gu@sydney.edu.au, ssw20@mails.tsinghua.edu.cn,
{chao.dong, lb.xie}@siat.ac.cn, xintaowang@tencent.com
yang.yujiu@sz.tsinghua.edu.cn

Abstract

The alignment of adjacent frames is considered an essential operation in video super-resolution (VSR). Advanced VSR models, including the latest VSR Transformers, are generally equipped with well-designed alignment modules. However, the progress of the self-attention mechanism may violate this common sense. In this paper, we rethink the role of alignment in VSR Transformers and make several counter-intuitive observations. Our experiments show that: (i) VSR Transformers can directly utilize multi-frame information from unaligned videos, and (ii) existing alignment methods are sometimes harmful to VSR Transformers. These observations indicate that we can further improve the performance of VSR Transformers simply by removing the alignment module and adopting a larger attention window. Nevertheless, such designs will dramatically increase the computational burden, and cannot deal with large motions. Therefore, we propose a new and efficient alignment method called patch alignment, which aligns image patches instead of pixels. VSR Transformers equipped with patch alignment could demonstrate state-of-the-art performance on multiple benchmarks. Our work provides valuable insights on how multi-frame information is used in VSR and how to select alignment methods for different networks/datasets. Codes and models will be released at https://github.com/XPixelGroup/RethinkVSRAlignment.

1 Introduction

Video Super-Resolution (VSR) could provide more accurate SR results than Single-Image Super-Resolution (SISR) by exploiting the complementary sub-pixel information from multiple frames. The well-established paradigms of VSR networks usually include an alignment module to compensate for the motion of objects between frames. The alignment module is critical for CNN-based VSR networks, because the locality inductive bias of CNNs only allows them to utilize spatial-close distributed information effectively. Many VSR networks have achieved better performance by introducing more advanced alignment methods [44, 4, 40, 24].

Recently, the general paradigm of vision network design has gradually shifted from CNNs to Transformers [10, 23, 7]. Unlike the locality property of CNNs, the self-attention operation in Transformers is very efficient for processing elements with spatially long-term distribution. VSR is no exception: Transformers have also been introduced into VSR for better results [3, 22, 24]. However,

[*] Jinjin Gu and Shuwei Shi contribute equally to this work.
[†] Corresponding author.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

these VSR Transformers still retain complex alignment modules. The ability of Transformers to efficiently process non-local information has not yet been exploited for inter-frame information processing.

In this paper, we rethink the role of the alignment module in VSR Transformers. We make two arguments: **(i)** *VSR Transformers can directly utilize multi-frame information from unaligned videos*, and **(ii)** *continuing to use the existing alignment methods will sometimes degrade the performance of VSR Transformers*. For the first argument, we report that for Transformers using the shifted window mechanism [31, 23], misalignment within a certain range does not affect the SR performance, while alignment only has a positive effect on the pixels beyond that range. The attribution map visualization results [11] also show that Transformers without alignment modules behave similarly to CNNs with alignment modules. For the second argument, we quantitatively compare the effects of various alignment methods. Using alignment will only have negative effects for pixels without large misalignment (*e.g.*, Vimeo-90K [48]). We blame this phenomenon on the noise of optical flow and the destruction of sub-pixel information by alignment resampling operations. Although implicit alignment based on deformable convolution can minimize these negative effects, the additional cost of parameters and computation makes this approach no longer advantageous.

According to our findings, we can build an alignment-free VSR Transformer, which requires a large attention window (>16) to cover misalignment between frames. However, enlarging the window size will lead to higher computational costs, making this approach no longer feasible. As a better alternative, we propose a simple yet effective alignment method called Patch Alignment, which aligns image patches instead of pixels. It will find the corresponding patches in the supporting frames and compute self-attention among them. This method uses a simple crop-then-move strategy to compensate for situations where the misalignment exceeds the Transformer's attention window. Experiments show that the proposed method achieves state-of-the-art VSR performance with a simple design and fewer parameters.

2 Related Works

VSR networks. The uniqueness of the VSR task lies in the utilization of inter-frame sub-pixel information [30]. Due to the motion of objects between frames, the computation and compensation for this motion is often the focus of VSR research. The pipeline of most VSR networks mainly includes an alignment module, a feature extraction module, a fusion module, and a reconstruction module. These networks are trained using a large number of low-resolution (LR) and high-resolution (HR) video sequences. The paradigm of early methods [25, 15, 29, 16, 48, 14, 27] can be summarized as: first estimate and compensate object motion (using optical flow), then pass through different feature extraction [37, 47, 19], fusion [39, 2, 4], and reconstruction modules [38], respectively. These methods can be further divided into methods for compensating motion in the image space (image alignment) and that in the feature space (feature alignment) [4, 43]. As Chan *et al.*[4] claims, feature alignment can achieve better results because inaccuracy in flow estimation may be mitigated after feature extraction.

However, the above methods generally suffer from inaccurate optical flow estimation and defective alignment operation, resulting in noticeable artifacts in the aligned images [40]. With the invention of Deformable convolutions [9], CNN networks also have the ability to model geometric transformations. Deformable convolutions gradually replace explicit warping operations in VSR networks with their learnable and flexible properties [44, 52, 40]. EDVR [44] integrates deformable convolution in the VSR network for the first time and aligns features at multiple scales. Chan *et al.*[5] incorporate optical flow as a guidance for offsets learning. The latest BasicVSR++ [6] based on the bidirectional loop structure also uses the alignment based on deformable convolution.

Differently, our paper aims to study Transformers' ability to model multi-frame information implicitly without using alignment. Before Transformer, attempts have been made to achieve the same effect in CNN, *i.e.*, the VSR methods without alignment. Both 2D [33, 49] and 3D convolutions [41, 13, 20, 16] are used to extract correlations among frames. Similar to Transformers, some methods use correspondence calculation to build inter-frame connections without alignment. For example, MuCAN [21] uses a temporal multi-correspondence aggregation module and a cross-scale non-local-correspondence aggregation module to replace explicit alignment. Yi *et al.*[51] introduce an improved non-local operation to avoid the complex alignment procedure. Generally speaking, models without explicit alignment are less effective or rely on special module design. However, the Transformers discussed in this paper could efficiently handle unaligned frames without additional special design.

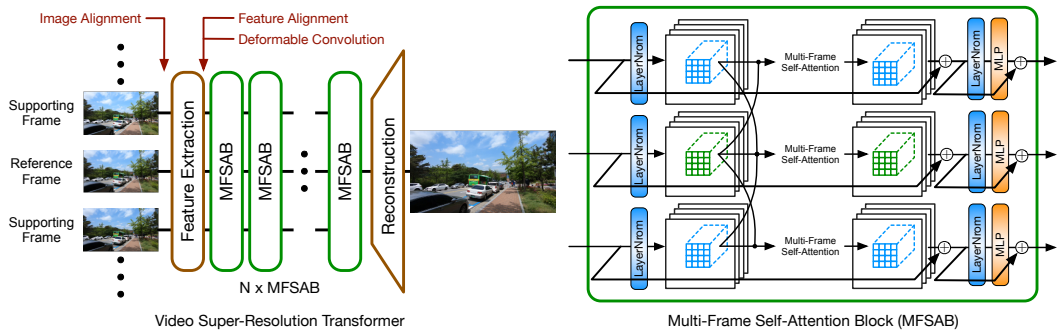

Figure 1: A brief illustration of the VSR transformer network used. The illustrated structure is based on the sliding window mechanism. We mark the position of the existing alignments methods.

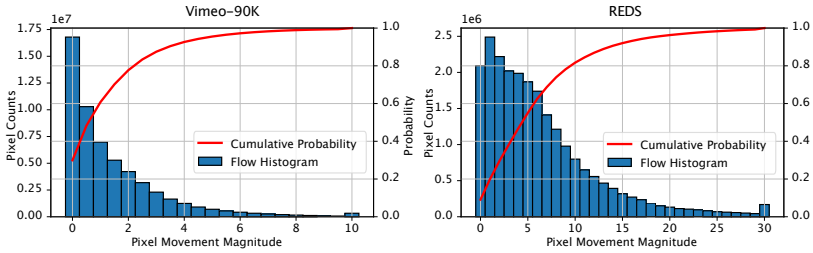

Figure 2: The distribution of the movement for the Vimeo-90K [48] and REDS [34] test sets. The distribution of pixel movement exhibits a long-tailed feature. The REDS dataset contains a larger movement.

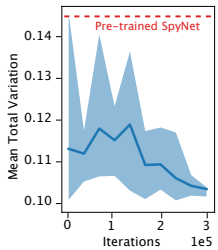

Figure 3: The curve of total variation of the fine-tuned flow.

Transformer networks. Transformers [42] are a new type of networks characterized by self-attention operations. A Transformer is considered to have a parameter-independent global receptive field within the scope of self-attention. Dosovitskiy *et al.*[10] first introduces Transformer into image recognition by projecting large image patches into a sequence of tokens. Inspired by the success of vision Transformers, many attempts have been made to employ Transformers for low-level vision tasks [50, 53, 7, 46, 54, 8] As a representative model, Liang *et al.*proposes SwinIR [23] that integrates a powerful shifted window strategy and achieves promising results. Transformer-based models have also been proposed for VSR. Cao *et al.*[3] employ the spatial-temporal self-attention on multi-frame patches to perform implicit motion compensation. Liang *et al.*[22] propose a parallel frame prediction Transformer to model long-range temporal dependency. Both the above Transformers consist of alignment modules. VSRT [3] uses feature alignment method and VRT [22] employs complex deformable convolution to perform forward and backward alignment. In this paper, we show that these complex alignment operations are unnecessary or even harmful for VSR Transformers.

3 Preliminary Settings

In this section, we describe the settings used in our experiments in detail, including the Transformer architecture, alignment methods, datasets, metrics and implementation details. If the readers are already familiar with the common VSR settings and VSR networks, it is okay to skip this section and go to Section 4 for experimental results and analysis. Due to space constraints, we record the detailed setup of the different experiments in Appendix G to facilitate the readers to reproduce our results.

The VSR Transformer architecture. We first describe the basic VSR Transformer backbone used in this study, which follows the sliding window design. Transformer based on shifted window mechanism [23] is proven to be flexible and effective for image processing tasks. We only make minimal changes when applying it to VSR to maintain its good performance without the loss of generality. This model is illustrated in Figure 1. The VSR Transformer takes a reference frame I^t and $2n$ adjacent supporting frames $\{I^{t-n}, \ldots, I^{t-1}, I^{t+1}, \ldots, I^{t+n}\}$ as input, and outputs the super-resolved reference frame I^t_{SR}. In the beginning, a feature extraction module is used to extract features for the subsequent Transformer. We use a single 2D convolution layer for feature extraction. The extracted features are denoted as $2n+1$ feature maps $\{X^{t-n}, \ldots, X^t, \ldots, X^{t+n}\}$.

Then N Multi-Frame Self-Attention Blocks (MFSAB) are used as the backbone network. The MFSAB is modified from RSTB in SwinIR [23], which contains two LayerNorm [1] layers, a multi-head self-attention layer and a multi-layer perceptron layer. We mainly modify the self-

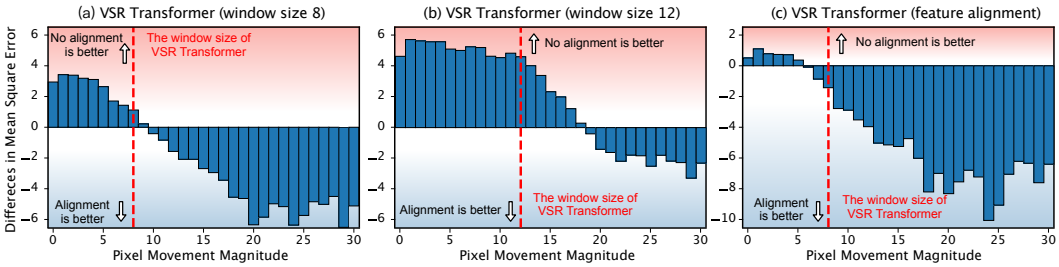

Figure 4: This figure illustrates the performance differences between VSR Transformers with and without alignment module for different pixel movement conditions. Parts greater than zero indicate better performance without alignment.

attention layer to make it suitable for the VSR task. Given these $2n+1$ feature maps of the frames with size $H \times W \times C$ (H, W and C are the height, width and feature dimension), the shifted window mechanism first reshapes the feature maps of each frame to $\frac{HW}{M^2} \times M^2 \times C$ features by partitioning the input into non-overlapping $M \times M$ local windows, where $\frac{HW}{M^2}$ is the total number of windows. We calculate self-attention on the features in the windows corresponding to the positions in different frames. Therefore, $(2n+1)M^2$ features are involved in each standard self-attention operation, and we concatenate features from different frames to produce a local window feature $X \in \mathbb{R}^{(2n+1)M^2 \times C}$. In each self-attention layer, the query Q, key K and value V are computed as $Q = XW^Q, K = XW^K, V = XW^V$, where $W^Q, W^K, W^V \in \mathbb{R}^{C \times D}$ are weight matrices, and D is the channel number of projected vectors. Then, we use Q to query K to generate the attention map $A = \texttt{softmax}(QK^T/\sqrt{D} + B) \in \mathbb{R}^{(2n+1)M^2 \times (2n+1)M^2}$, where B is the learnable relative positional encoding. This attention map A is then used for the weighted sum of $(2n+1)M^2$ vectors in V. The multi-head settings are aligned with SwinIR [23] and ViT [10]. In the final reconstruction module, only the features of the reference frame are used to generate the SR result. We use the pixel-shuffle layer [38] to upsample the feature maps. Besides sliding window based VSR Transformer, the MFSAB can also be applied to recurrent frameworks [6]. We describe the details in Appendix A.

Alignment methods. Alignment methods follow a long line of research. To investigate the role of alignment, we need to implement and compare different alignment methods. These alignment methods can be classified into four types, and their respective representative methods are included in our experiments. We next describe their characteristics and then describe our implementation.

- *Image alignment* is the earliest and most intuitive alignment method, which is also the easiest to use. Image alignment relies on the explicitly calculated optical flow between frames. According to the estimated inter-frame motion, different frames are aligned by a warping operation. Following the existing successful experience [48, 4], we use the SpyNet [36] to estimate the optical flow, and fine-tune the SpyNet simultaneously during training. The resampling method used is the bilinear (BI) method.
- *Feature alignment* also estimates the optical flow but performs the warping operation on the deep features instead of on the images. The flow estimation module still uses SpyNet, which is optimized during training. In addition to the shallow 2D convolution in Figure 1, we add five additional residual blocks [12] to extract deep features.
- *Deformable convolution (DC) based methods* use learnable dynamic deformable convolution for alignment. Almost all state-of-the-art VSR networks use deformable convolution to perform the alignment. We employ the flow-guided deformable convolution (FGDC) alignment used in BasicVSR++ [6] and VRT [22] as the representative method.
- *No alignment.* The raw input is processed directly using the VSR Transformer.

Datasets and metrics. In the VSR literature, the REDS [34] and the Vimeo-90K [48] datasets are the de-facto benchmarks. REDS has 270 available video sequences, each containing 100 frames. We follow the common splitting methods and split the data into training (266 sequences) and testing (4 sequences) sets. Vimeo-90K contains 64,612 and 7,824 video sequences for training and testing, respectively. Although REDS and Vimeo-90K are widely used benchmarks, these two datasets have different motion conditions. The motion in the Vimeo-90K dataset is generally small, and movement magnitudes of 99% pixels are less than 10 (for each clip, we measure the motion of the 4th and the 7th frames). Differently, there are large motions in the REDS dataset. There are at least 20% pixels that have movement magnitudes larger than 10 (for each clip, we measure the motion of the 3rd and the 5th frames). The distributions of the movement conditions for these two datasets are shown in

Table 1: Quantitative comparison of different VSR methods. The results marked with $*$ achieve similar performance as no alignment. This is due to the vanishing of optical flow in this experiment. Details are shown in Figure 5 and discussed in Section 4.2

Exp. Index	Method	Alignment	Remark	Vimeo90K-T		REDS4	
				PSNR	SSIM	PSNR	SSIM
1	VSR-CNN	Image alignment	Finetune flow	36.13	0.9342	29.81	0.8541
2	VSR-CNN	No alignment		36.24	0.9359	28.95	0.8280
3	VSR Transformer	Image alignment	Fix flow	36.87	0.9429	30.25	0.8637
4	VSR Transformer	Image alignment	Finetune flow	37.44*	0.9472*	30.43	0.8677
5	VSR Transformer	Feature alignment	Finetune flow	37.36	0.9468	30.74	0.8740
6	VSR Transformer	No alignment	Window size 8	37.43	0.9470	30.56	0.8696
7	VSR Transformer	No alignment	Window size 16	37.46	0.9474	30.81	0.8745

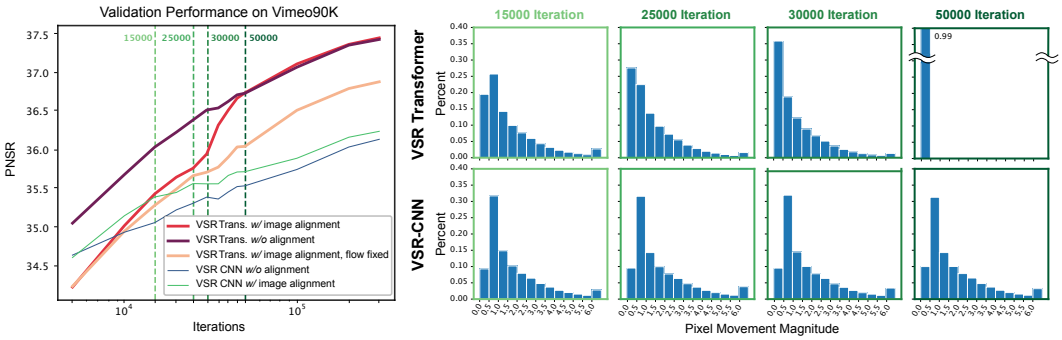

Figure 5: The figure on the left shows the validation curves tested on Vimeo-90K test set. On the right are the histogram distribution of the estimated optical flow at different training iterations. The moment when the optical flow distribution changes corresponds to the moment when the performance of the VSR Transformer with alignment increases.

Figure 2. In our experiments, we mainly study $\times 4$ VSR task. We use bicubic interpolation to produce LR video frames. Peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) are used for evaluation.

Implementation. We implement all these methods using the BasicSR framework [45]. We use the Charbonnier loss [18] as the training objective. The Adam optimization [17] method is used for training with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is set to 4×10^{-4}, and a Consine Annealing scheme [32] is used to decay the learning rate. The total iteration number is set to 300,000. The mini-batch size is 8, and the LR patch size is 64×64. The experiments are implemented based on PyTorch [35] and conducted on NVIDIA A100 GPUs. The details and configurations for each experiment can be found in Appendix G.

4 Rethinking Alignment

In this section, we conduct experiments based on the above preliminary settings. We use four questions to guide our rethinking. We present our experimental results and analysis in each subsection.

4.1 Does alignment always benefit VSR Transformers?

In previous VSR research, we often judge the pros and cons of the method based on the average performance on the test set. But for the VSR Transformers, the performance of alignment inside the local window is different from that outside the window because of the limited range of self-attention. A finer perspective on how different methods perform on different data may help us understand the effect of alignment. We investigate the performance difference between the VSR Transformer with and without alignment for different pixel movement conditions. Figure 4 shows the results tested on the REDS dataset and Table 1 reports the quantitative results for these experiments.

One can draw the following observations. First, as can be observed from Figure 4 (a), for pixels with small movements, VSR Transformer can achieve good results *without* alignment. Using image alignment in these cases brings negative effects. This range of pixel movement is related to the window size used by the VSR Transformer. As there is no locality inductive bias in the processing of pixels within a local window, the Transformer can handle misalignment in this range. Second, as the movement increases, the information required by VSR exceeds the scope of the local windows. In these cases, image alignment can improve performance. Third, according to Figure 2, the movement

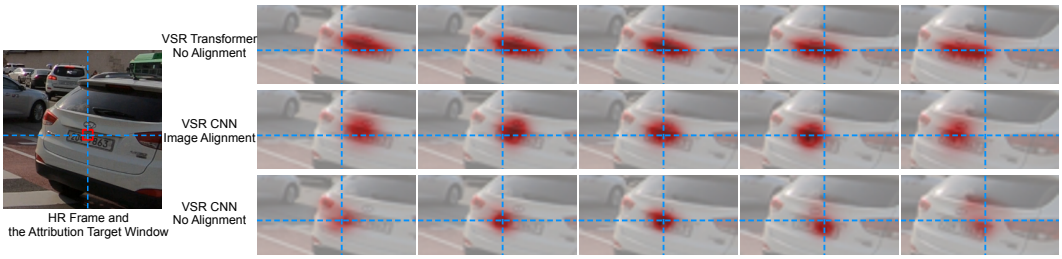

Figure 6: The attribution results of different models. We use the local attribution map [11] to highlight the responsible pixels in each frame for the SR of the selected red window.

Table 2: Comparison of VSR Transformers with different alignment methods on the REDS4 dataset.

#	Alignment Method				Position		Resampling		Params.	REDS4
	No Ali.	Img. Ali.	Feat. Ali.	FGDC	Img.	Feat.	BI	NN	(M)	PSNR / SSIM
1	✓								12.9	30.92 / 0.8759
2		✓			✓		✓		12.9	30.84 / 0.8752
3			✓			✓	✓		14.8	31.06 / 0.8792
4			✓			✓		✓	14.8	31.11 / 0.8801
5				✓		✓			16.1	31.11 / 0.8804

of about 70% pixels is less than eight on the REDS test set. This makes the model without alignment still perform better than the image alignment on this test set.

We then increase the size of the shifted window, and conduct the same experiment. The result is shown in Figure 4 (b). We can see that VSR Transformer can handle a larger range of unaligned pixels with a larger window size. This shows that the processing capability of VSR Transformers for unaligned frames is related to the size of the shifted window, and also implies that this capability mostly relies on the self-attention mechanism. To investigate whether better alignment methods can eliminate the negative effects of alignment on small motion pixels, we use feature alignment to conduct the same experiment, and the result is shown in Figure 4 (c). As one can see, feature alignment narrows the gap between alignment and no alignment on VSR Transformer, but it still has negative effects on the pixels of small motion.

In summary, the VSR Transformer can handle misalignment within a certain range, and using alignment at this range will bring negative effects. This range is closely related to the window size of the VSR Transformer. However, alignment is necessary for motions beyond the VSR Transformer's processing range.

4.2 What kinds of flow are better for VSR?

Although using optical flow for alignment can have a negative impact, different flows can also lead to differences in performance. This inspires us to think about what kinds of flow are better for VSR? According to Table 1, one can see that optimizing the flow estimator while training the VSR network will bring better results, because the flow estimator at this time learns the optimized flow for VSR [48]. The property of this task-oriented flow can imply how the VSR Transformers use multi-frame information. Our first observation is that VSR Transformers tend to use smooth flow. The flow estimator SpyNet was pre-trained with the end-point-error (EPE) loss, which does not explicitly encourage smoothness. The non-smooth flow will introduce random noise to VSR and lose sub-pixel information. To study this problem, we investigate the change of the estimated flow while training the VSR Transformer with image alignment using the REDS dataset. We can see that the flow estimated by the fine-tuned SpyNet is getting smoother, which is reflected in the decrease of the average total variation, as shown in Figure 3. Smoother flow maintains the relative relationship of adjacent pixels in the aligned frames, thus facilitating VSR processing.

Although the fine-tuned flow estimator will improve performance, there is still a gap between image alignment with flow fine-tuning and no alignment on the REDS dataset. However, we observe different results on the Vimeo-90K dataset: image alignment with flow fine-tuning is almost identical to no alignment. By observing the distribution of the estimated flow, we are surprised to find that the flow is slowly decreasing to 0 when fine-tuned with image alignment using Vimeo-90K. This is why the final result is almost the same as no alignment. In Figure 5, we show the validation curve of the related experiments and the histogram distribution of the fine-tuned flow. As one can see, there is a large gap between the VSR Transformers with and without alignment in the early stage of training. When the training reaches 25,000 iterations, the fine-tuned flow gradually shifts towards the zero

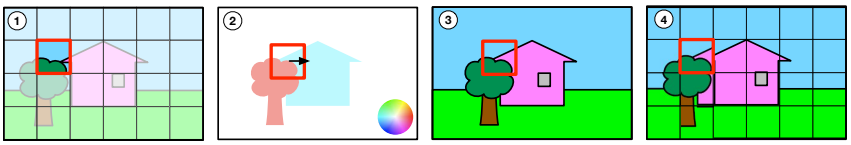

Figure 7: The pipeline of the proposed Patch Alignment method: ① partition the input frames to patches according to the window partition of Transformer, ② calculate the mean motion vector for each patch, ③ find the corresponding patches in the supporting frames, and ④ move the entire supporting patches to their corresponding position.

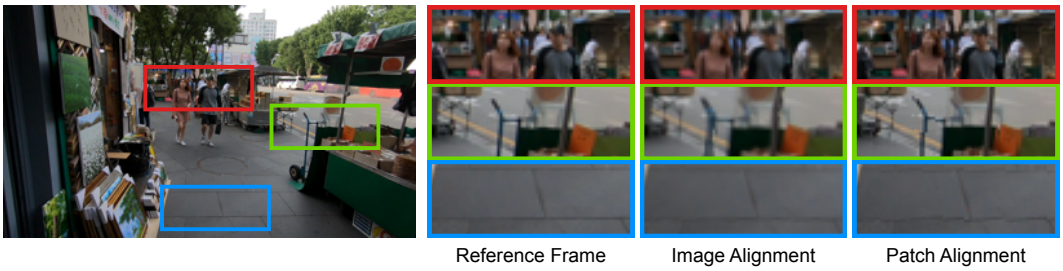

Figure 8: The visualization comparison of image alignment and the proposed patch alignment.

direction. When training after 50,000 iterations, the fine-tuned flow vanishes. At this time, the VSR Transformer with alignment is equivalent to the one without alignment. This phenomenon does not appear on VSR-CNN. This experiment is instructive. On the one hand, most of the movements in the Vimeo-90K dataset are smaller than the Transformer's window size. According to Figure 4 (a), alignment is detrimental to training at this situation. The fine-tuned flow estimator seems aware of this knowledge and learns to improve performance by forcing flow value to all zeros. On the other hand, the adaptability of the model is surprising. The model will choose the policy that maximizes the training objective, even if that policy completely disables part of the network.

We can now answer the question posed by this subsection. For most VSR Transformers, the smooth flow is better if the alignment is necessary. For situations with small motions, the flow with all zeros is the best for VSR Transformers.

4.3 Does Transformer implicitly track the motion between unaligned frames?

We already know that VSR Transformers can handle a specific range of misalignment, but do they track these movements implicitly? Can an alignment-like function be done inside the VSR Transformers? We next use an interpretability tool to visualize the behaviour of the VSR Transformer. Local Attribution Map (LAM) [11] is an attribution method aiming to find input pixels strongly influencing network output. We first specify a target patch on the output image and then use LAM to generate the corresponding attribution maps. We will track the information used by the model, and see which parts of pixels in adjacent frames contribute the most. As objects move between frames, an ideal VSR network needs to track those movements and utilize pixels representing the same object. We show some representative results in Figure 6. It can be observed that even without the alignment module, the VSR Transformer can automatically change its attention to the most relevant pixels. Intuitively, the self-attention operation can detect the associations between pixels in different frames and utilize their additional information for VSR. This is similar to VSR CNNs with the alignment module. We also show results of VSR CNNs without the alignment module. Since CNN has the inductive bias of locality, it can only focus on the area near the current position if no alignment is used. These areas are less valuable to VSR. Thus, alignment is an essential module for CNN-based VSR methods.

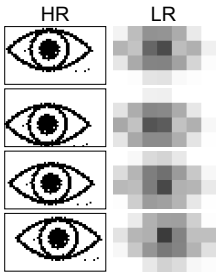

Figure 10: Illustration of sub-pixel information.

4.4 Why do alignment methods have negative effects?

To understand the reasons for the negative impact of alignment, we need to know what sub-pixel information does VSR require. As shown in Figure 10, high-frequency information in HR frames is lost during downsampling, and only aliasing patterns are left in the LR frames. When the HR frames move, different LR frames are generated, and different aliasing patterns are produced. These

Method	Position		Resampling		REDS4	
	Img.	Feat.	BI	NN	PSNR	SSIM
Patch Alignment	✓			✓	31.11	0.8800
		✓	✓		31.00	0.8781
		✓		✓	31.17	0.8810

Table 3: The Ablation study of Patch Alignment. We study the effect of different resampling methods (BI and NN) and different alignment positions (image space and feature space).

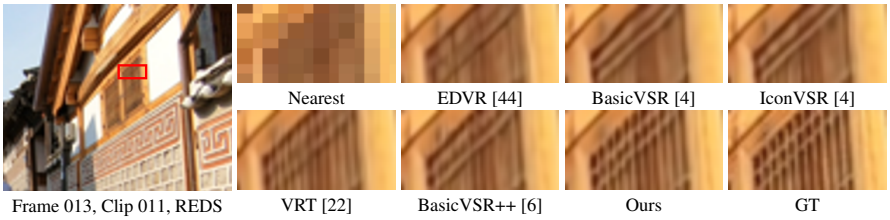

Figure 9: Visual comparison of VSR ($\times 4$) on REDS dataset.

patterns provide additional constraints for VSR. However, the inaccurate optical flow and the bilinear resampling operation could corrupt these patterns. First, the inaccurate flow can be viewed as a combination of the ground-truth flow and a random error term. Using such flow for alignment randomly will change the LR patterns and cause information loss. Second, the bilinear resampling operation calculates the weighted average among four adjacent pixels, and the weights are inaccessible for the VSR model. At this time, the interpolation is irreversible. The VSR model can only process the transformed LR patterns and cannot access the original patterns, resulting in information loss. We conduct experiments to demonstrate the negative effects of optical flow and resampling methods. The results are shown in Table 2. Compared with image alignment, feature alignment improves performance by extracting part of the sub-pixel information before it is corrupted by alignment. The flow-guided deformable convolution (FGDC) reduces the negative effects of alignment by enabling the network to model geometric transformations. Changing the resampling method to nearest neighbor (NN) also improves performance because the NN method can preserve the relationship between adjacent pixels and ignore the noise of flow estimation to a certain extent. As can be seen, feature alignment using the NN resampling method achieves the same performance as the FGDC method but with a significantly reduced number of parameters.

5 Patch Alignment

According to our observations, an inaccurate flow and the resampling operation will impair the utilization of the inter-frame information. To overcome this problem, simply increasing the Transformer's window size is a straightforward solution as Transformers are naturally good at modelling unaligned spatial dependencies within the local window. However, the time and space complexity of increasing the window size is at least $\mathcal{O}(n^2)$. We need more efficient alignment methods to improve performance for large movement pixels while introducing little negative impact and additional computational complexity. In this section, we propose a simple yet effective alignment method for VSR Transformers, called Patch Alignment.

5.1 Method

The pipeline of the proposed method is shown in Figure 7. This method does not align individual pixels but treats the image as non-overlapping patches. The partition of patches is consistent with the partition of the Transformer's local windows. We treat the patch as a whole and perform the same operation on the pixels within the patch. In this way, the relative relationship between pixels is kept, and resampling operations will not corrupt the sub-pixel information within the patch. We locate the movement of objects based on optical flow, yet we do NOT pursue precise pixel-level alignment. We calculate the mean motion vector in each patch and find the corresponding patches in the supporting frames for each patch. Next, we use the nearest neighbor resampling method to move the entire supporting patches to their corresponding position in the reference frame. The benefits are three-fold. First, the nearest neighbor resampling ignores the fractional part of optical flow estimation and reduces the error caused by inaccurate flow estimation. Second, the entire patch is cropped and moved to the corresponding position, which preserves the relative relationship of the pixels within the patch and thus retains the sub-pixel information.

We show the comparison of image alignment with bilinear resampling and the proposed patch alignment method in Figure 8. As can be seen, image alignment introduces blurry and artifacts to

Table 4: Quantitative comparison (PSNR↑ and SSIM↑) on the REDS4 [34] dataset, Vid4 [28], Vimeo-90K-T [48] dataset for $4\times$ VSR task. Red indicates the best and blue indicates the second best performance (best view in color) in each group of experiments.

Method	Frames REDS/Vimeo	Params (M)	REDS4 PSNR	REDS4 SSIM	Vimeo-90K-T PSNR	Vimeo-90K-T SSIM	Vid4 PSNR	Vid4 SSIM
EDVR [44]	5/7	20.6	31.09	0.8800	37.61	0.9489	27.35	0.8264
VSR-T [3]	5/7	32.6	31.19	0.8815	37.71	0.9494	27.36	0.8258
PSRT-sliding	5/-	14.8	31.32	0.8834	-	-	-	-
VRT	6/-	30.7	31.60	0.8888	-	-	-	-
PSRT-recurrent	6/-	10.8	31.88	0.8964	-	-	-	-
BasicVSR [4]	15/14	6.3	31.42	0.8909	37.18	0.9450	27.24	0.8251
IconVSR [4]	15/14	8.7	31.67	0.8948	37.47	0.9476	27.39	0.8279
BasicVSR++ [6]	30/14	7.3	32.39	0.9069	37.79	0.9500	27.79	0.8400
VRT	16/7	35.6	32.19	0.9006	38.20	0.9530	27.93	0.8425
PSRT-recurrent	16/14	13.4	32.72	0.9106	38.27	0.9536	28.07	0.8485

the aligned image that destroy sub-pixel information. Patch alignment retains more details that can provide additional information for the VSR model. As we do not pursue pixel-level alignment, directly operating on patches will leave discontinuous artifacts along patch borders. But our experiments show that these discontinuities have little effect on VSR Transformers. Because these discontinuities do not appear in the local window of the Transformer, they do not affect the function of self-attention. It further illustrates the importance of preserving sub-pixel information.

5.2 Experimental results

We test the patch alignment method with different alignment positions and resampling methods. The experimental settings and the network configurations are the same with experiments in Table 2. The results are shown in Table 3. As can be seen, image-level patch alignment is already comparable to FGDC, while the latter uses 25% more parameters. Feature-level patch alignment achieves the best performance among all the tested alignment methods. We conduct an ablation experiment using bilinear resampling to study the importance of NN resampling for patch alignment. It can be seen that bilinear resampling leads to a severe performance drop. It shows that the LR patterns retained by the NN method are critical for VSR.

We use patch alignment to build VSR Transformers based on the sliding window and the recurrent frameworks, namely PSRT-sliding and PSRT-recurrent. We test their performances on REDS and Vimeo-90K datasets. The results are shown in Table 4. VSR Transformers with patch alignment can achieve state-of-the-art performance with fewer parameters compared to other Transformer-based VSR methods (VSRT [3] and VRT [22]). Compared to BasicVSR++ [6], we use fewer frames in training, but achieve a 0.33dB improvement on REDS in terms of PSNR. Visual results of different methods are shown in Figure 9. As one can see, in accordance with its significant quantitative improvements, the proposed method can generate visually pleasing images with sharp edges and fine details. By contrast, its competitors suffer from either blurry or lost details.

6 Conclusion

In this paper, we present two important conclusions about using Transformers in the VSR task: (i) VSR Transformers can directly utilize multi-frame information from unaligned videos, and (ii) existing alignment methods are sometimes harmful to VSR Transformers. We also propose a new patch alignment method for VSR Transformers. The proposed method demonstrates the state-of-the-art performance for VSR.

Given the current literature, our results are interesting and inspiring. They challenge our common understanding of using Transformers to process multiple spatially misaligned images. First, the analysis of alignment can provide useful insights for VSR. We need to utilize inter-frame sub-pixel information, yet many image pre-process operations interfere with our utilization of this information. Second, these observations hint that Transformer can implicitly make accurate connections for misaligned pixels. Many low-level vision tasks can take advantage of this property, such as video restoration, reference-based SR, burst image processing, stereo matching, *etc*. When designing Transformers for these tasks, we can no longer explicitly introduce the alignment modules or the cost volume modules, but give full play to the powerful modeling capabilities of the Transformer itself.

Acknowledgement This work was supported in part by the National Natural Science Foundation of China (Grant No. 61991450) and the Shenzhen Key Laboratory of Marine IntelliSense and Computation (ZDSYS20200811142605016).

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Wenbo Bao, Wei-Sheng Lai, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Memc-net: Motion estimation and motion compensation driven neural network for video interpolation and enhancement. *IEEE transactions on pattern analysis and machine intelligence*, 43(3):933–948, 2019.

[3] Jiezhang Cao, Yawei Li, Kai Zhang, and Luc Van Gool. Video super-resolution transformer. *arXiv preprint arXiv:2106.06847*, 2021.

[4] 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*, 2021.

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

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

[7] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *CVPR*, 2021.

[8] Zheng Chen, Yulun Zhang, Jinjin Gu, Yongbing Zhang, Linghe Kong, and Xin Yuan. Cross aggregation transformer for image restoration. In *NIPS*, 2022.

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

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[11] Jinjin Gu and Chao Dong. Interpreting super-resolution networks with local attribution maps. In *CVPR*, 2021.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[13] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *CVPR*, 2018.

[14] Ratheesh Kalarot and Fatih Porikli. Multiboot vsr: Multi-stage multi-reference bootstrapping for video super-resolution. In *CVPRW*, 2019.

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

[16] Tae Hyun Kim, Mehdi SM Sajjadi, Michael Hirsch, and Bernhard Scholkopf. Spatio-temporal transformer network for video restoration. In *ECCV*, 2018.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2014.

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

[19] Dingyi Li, Yu Liu, and Zengfu Wang. Video super-resolution using non-simultaneous fully recurrent convolutional network. *IEEE Transactions on Image Processing*, 28(3):1342–1355, 2018.

[20] Sheng Li, Fengxiang He, Bo Du, Lefei Zhang, Yonghao Xu, and Dacheng Tao. Fast spatio-temporal residual network for video super-resolution. In *CVPR*, 2019.

[21] Wenbo Li, Xin Tao, Taian Guo, Lu Qi, Jiangbo Lu, and Jiaya Jia. Mucan: Multi-correspondence aggregation network for video super-resolution. In *ECCV*, 2020.

[22] 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.

[23] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. In *CVPR*, 2021.

[24] Jingyun Liang, Yuchen Fan, Xiaoyu Xiang, Rakesh Ranjan, Eddy Ilg, Simon Green, Jiezhang Cao, Kai Zhang, Radu Timofte, and Luc Van Gool. Recurrent video restoration transformer with guided deformable attention. *arXiv preprint arXiv:2206.02146*, 2022.

[25] Renjie Liao, Xin Tao, Ruiyu Li, Ziyang Ma, and Jiaya Jia. Video super-resolution via deep draft-ensemble learning. In *ICCV*, 2015.

[26] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017.

[27] Jing Lin, Yuanhao Cai, Xiaowan Hu, Haoqian Wang, Youliang Yan, Xueyi Zou, Henghui Ding, Yulun Zhang, Radu Timofte, and Luc Van Gool. Flow-guided sparse transformer for video deblurring. In *ICML*, 2022.

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

[29] Ding Liu, Zhaowen Wang, Yuchen Fan, Xianming Liu, Zhangyang Wang, Shiyu Chang, and Thomas Huang. Robust video super-resolution with learned temporal dynamics. In *ICCV*, 2017.

[30] Hongying Liu, Zhubo Ruan, Peng Zhao, Chao Dong, Fanhua Shang, Yuanyuan Liu, Linlin Yang, and Radu Timofte. Video super-resolution based on deep learning: a comprehensive survey. *Artificial Intelligence Review*, pages 1–55, 2022.

[31] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[32] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

[33] Alice Lucas, Santiago Lopez-Tapia, Rafael Molina, and Aggelos K Katsaggelos. Generative adversarial networks and perceptual losses for video super-resolution. *IEEE Transactions on Image Processing*, 28(7):3312–3327, 2019.

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

[35] 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. In *NIPS*, 2019.

[36] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *CVPR*, 2017.

[37] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *CVPR*, 2018.

[38] 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*, 2016.

[39] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *CVPR*, 2017.

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

[41] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[43] Hua Wang, Dewei Su, Chuangchuang Liu, Longcun Jin, Xianfang Sun, and Xinyi Peng. Deformable non-local network for video super-resolution. *IEEE Access*, 7:177734–177744, 2019.

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

[45] Xintao Wang, Ke Yu, Kelvin C.K. Chan, Chao Dong, and Chen Change Loy. BasicSR: Open source image and video restoration toolbox. `https://github.com/xinntao/BasicSR`, 2018.

[46] Zhendong Wang, Xiaodong Cun, Jianmin Bao, and Jianzhuang Liu. Uformer: A general u-shaped transformer for image restoration. *arXiv preprint arXiv:2106.03106*, 2021.

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

[48] 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.

[49] Bo Yan, Chuming Lin, and Weimin Tan. Frame and feature-context video super-resolution. In *AAAI*, 2019.

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

[51] 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*, 2019.

[52] Xinyi Ying, Longguang Wang, Yingqian Wang, Weidong Sheng, Wei An, and Yulan Guo. Deformable 3d convolution for video super-resolution. *IEEE Signal Processing Letters*, 27:1500–1504, 2020.

[53] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In *CVPR*, 2022.

[54] Jiale Zhang, Yulun Zhang, Jinjin Gu, Yongbing Zhang, Linghe Kong, and Xin Yuan. Accurate image restoration with attention retractable transformer. *arXiv preprint arXiv:2210.01427*, 2022.

[55] 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.

Checklist

1. For all authors.
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [Yes]

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix

A Patch Alignment for Recurrent-based VSR Transformer

The proposed patch alignment method can also be applied to the recurrent VSR framework. Recurrent VSR methods [4, 6] use bidirectional propagation scheme to maximize information gathering in VSR and have achieved the state-of-the-art performance. By replacing the CNN backbone with the Transformer backbone, we can easily build a recurrent VSR Transformer. We employ the second-order grid propagation framework similar to BasciVSR++ [6], where the intermediate features are propagated both forward and backward in an alternating fashion. Through propagation, information from different frames can be used for feature refinement. We replace the feature propagation bock with the MFSAB blocks presented in the main text. The architecture of this recurrent VSR Transformer is shown in Figure 12.

Alignment modules are not absent in the existing recurrent methods. In each feature propagation block, features from different frames are aligned to extract information from the adjacent frames, improving feature expressiveness. BasicVSR [4] uses flow-based alignment method for both images and features and BasicVSR++ [6] uses flow-guided deformable convolution (FGDC) alignment. The proposed patch alignment is also compatible with this architecture. We test different alignment methods on the recurrent VSR Transformer; the results are shown in Table 5. In the recurrent VSR Transformers in this Table, we use 12 MFSABs with shortcut connections every 3 MFSABs for each feature propagation block. The feature size is set to 100, and the number of attention heads is 4. The baseline is the original BasicVSR++ model that uses FGDC and CNN backbone. Replacing the CNN with Transformer blocks can bring a PSNR improvement of 0.5dB on the REDS test set. However, the FGDC alignment used 7.8M parameters, accounting for almost half of all parameters. Replacing the FGDC alignment with the proposed patch alignment achieves competitive results without introducing additional parameters – our method saves 7.8M of parameters. This experiment illustrates the effectiveness of the proposed patch alignment method.

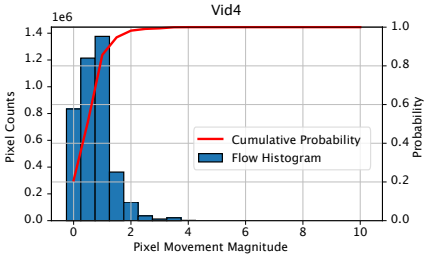

Figure 11: The distribution of the movement for the Vid4 [28] test sets.

In the main text, we report that the proposed PSRT-recurrent trained using 16 frames demonstrates the state-of-the-art performance on the VSR task, even compared with BasicVSR++, which was trained using 30 frames. We also tried using 30 frames when training PSRT-recurrent. The training curve is shown in Figure 13. It can be seen that PSRT-recurrent can still be greatly improved from more training frames. However, training with 30 frames takes much longer time than with 16 frames, which makes this method uneconomical. In Table 8 we also compare the state-of-the-art contemporaneous work RVRT [24]. When RVRT uses 30 frames for training, it can achieve similar performance to PSRT-recurrent when it was trained with 16 frames. This also demonstrates the superior performance of our method.

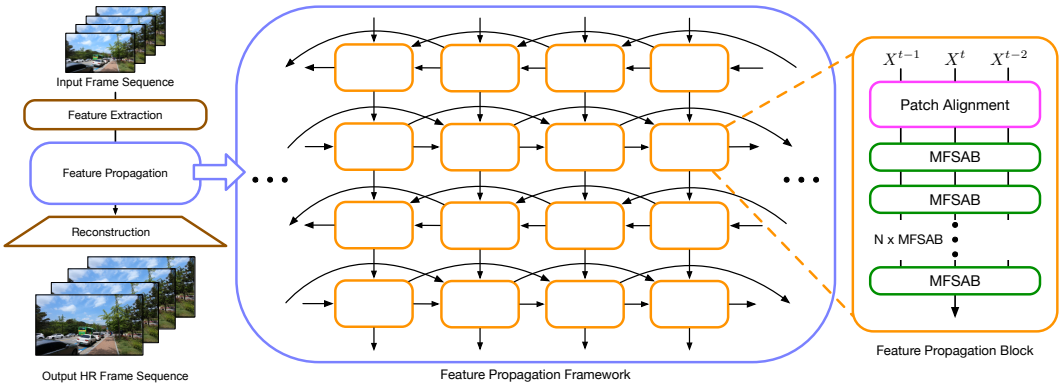

Figure 12: The framework of the used recurrent-based VSR Transformer.

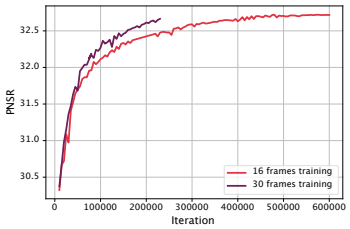

Figure 13: The comparison of with different training frames. Training with 30 frames leads to better performance at the cost of a larger training cost.

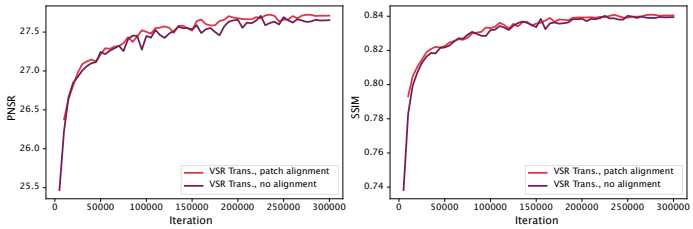

Figure 14: We compare the patch alignment method and no alignment on the Vimeo-90K dataset. In this case of small movements, no alignment can already achieve good performance, and patch alignment will not bring much improvement. This shows that Transformers can directly handle a small range of misalignment.

B Patch Alignment for the Vimeo-90K Dataset

Unlike the REDS [34] dataset, the motion in the Vimeo-90K dataset [48] tends to be smaller. According to Figure 2 in the main text, movement magnitudes of 98% pixels in the Vimeo-90K dataset are less than 8. As shown from Table 1 in the main text, VSR Transformer without alignment can outperform other alignment methods on this dataset. A natural question is whether the proposed patch alignment is still effective for a small-motion dataset. We conduct an ablation study on the Vimeo-90K and the Vid4 dataset. Table 6 reports the results. We also show the training curve of these two methods in Figure 14 and the distribution of pixel movement for the Vid4 [28] test set in Figure11. As can be observed from the distribution of movement, the Vid4 dataset contains no pixels whose movement magnitudes are larger than 5. Since most of the motion in these two datasets is within the range that VSR Transformer can handle, there is no significant difference between patch alignment and no alignment method, even no alignment version performs slightly better on Vimeo-90K. The validation curve also demonstrates that patch alignment and no alignment show comparable performance. This experiment confirms our conclusions that (1) we can get good results using VSR Transformers without additional alignment for a specific range of misalignment, and (2) the proposed patch alignment does not introduce as many negative effects as other alignment methods.

Finally, we present the results of recurrent VSR Transformer with patch alignment method trained using 14 frames in Table 8. This method achieves state-of-the-art performance on both the Vimeo-90K test set and the Vid4 test set.

C The FLOPs and Runtime of the Proposed Method

We calculate the average FLOPs of our method and some existing methods. This FLOPs is calculated using LR frames with size 180×320. We also record their average runtime. The results are shown in Table 7. As can be seen, the number of parameters of our method is less than other Transformer methods. One of the reasons is that our method saves a lot of parameters on the alignment module. Our FLOPs and runtime are also within a reasonable range. As the acceleration and optimization of Transformers are still to be studied, we believe that given our relatively small FLOPs, there is room for further optimization of the runtime of our method. For the training time, only VRT reports their

Table 5: Ablation study on the different alignment methods and backbone networks. The results are tested on REDS4 [34] dataset for $4\times$ video super-resolution on RGB channels.

Method	Frames	Params(M)	PSNR	SSIM
BasicVSR++ [6], The baseline model	6	7.3	31.38	0.8898
Flow-guided Deformable Alignment + Transformer	6	18.6	31.89	0.8967
Patch Alignment + Transformer	6	10.8	31.88	0.8964

Table 6: Ablation study of patch alignment method trained using the Vimeo-90K dataset. The results are tested on Vimeo-90K-T and Vid4 for $4\times$ video super-resolution on Y channel. The experiments with 7 training frames were only trained from scratch for 300K iterations.

Method	Frames	Params	Vimeo-90K-T		Vid4	
			PSNR	SSIM	PSNR	SSIM
PSRT-recurrent *w/o* alignment	7	12.0	37.87	0.9508	27.71	0.8403
PSRT-recurrent	7	13.4	37.80	0.9502	27.72	0.8409

training time. VRT need 15 days to train and the proposed PSRT-recurrent needs 18 days to train. Our method's training time and cost are roughly the same compared with VRT.

D Discussion about VSRT

We notice an exception in the effect of the alignment module to a VSR Transformer, *i.e.*, the VSRT model. Although the VSRT model [3] also employs Transformer as the backbone design, the alignment module is necessary for it. Removing alignment in VSRT introduces severe performance degradation. This conflicts with our conclusion. Our discussion on this issue is as follows. The VSRT uses a token size of 8×8. In the VSRT, self-attention is calculated between different tokens. This calculation is free of the indicative bias of locality. But within the 8×8 token, only convolution layers and MLP layers participate in the calculation. This calculation is subject to locality bias. If the 8×8 token is not well-aligned, the convolution layers and MLP layers cannot handle unaligned video frame tokens, and self-attention between tokens cannot help improve this. Therefore, the situation of VSRT does not conflict with the argument of this paper.

E Limitation

Our work discusses alignment in the VSR task, whose downsampling operation leads to unique LR patterns. We believe that other downsampling methods will have similar effects, such as blurring and directly downsampling (the "BD" downsampling in other papers). For these downsampling methods, the conclusions of this paper are still valid. Some of the observations may not apply to other video restoration tasks, because the multi-frame information they need to use may differ. For other video restoration tasks, we believe that the proposed method will still lead to improvement since theoretically patch alignment preserves more information. But if patch alignment is applied without modification, the resulted improvement may not be as big as in the VSR task. Because the nature of the sub-pixel information will change for other applications, the network design can also be changed (such as adding multi-scale designs). We only have limited space in this paper. However, we emphasise the importance of research in this direction and reserve it for future work.

F More Experiments

We show more visual comparisons between the existing VSR methods and the proposed recurrent VSR Transformer with the patch alignment method. We use 16 frames to train on the REDS dataset and seven on the Vimeo-90K dataset. Figure 15 shows the visual results. It can be seen that, in addition to its quantization improvement, the proposed method can generate visually pleasing images with sharp edges and fine details, such as horizontal bar patterns of buildings and numbers on license plates. In contrast, existing methods suffer from texture distortion or loss of detail in these scenes.

G Detail of Experiments

We present details of the experiments involved in this paper so that anyone can reproduce our results.

Figure 2 and Figure 11 illustrate the distribution of the movement for three datasets used in our work: Vimeo-90K test set [48], REDS [34] test set and Vid4 [28] test set. We use the pre-trained

Table 7: The comparison of the parameter numbers, FLOPs and the runtime for different methods.

Method	Parameters (M)	FLOPs (T)	Runtime (ms)
DUF	5.8	2.34	974
RBPN	12.2	8.51	1507
EDVR [44]	20.6	2.95	378
VSRT [3]	32.6	1.6	–
VRT [22]	35.6	1.3	243
PSRT-recurrent (Ours)	13.4	1.5	812

Table 8: Quantitative comparison (PSNR↑ and SSIM↑) on the REDS4 [34] dataset, Vid4 [28], Vimeo-90K-T [48] dataset for $4\times$ VSR task. Red indicates the best and blue indicates the second best performance (best view in color) in each group of experiments.

Method	Frames REDS/Vimeo	Params (M)	REDS4 PSNR	REDS4 SSIM	Vimeo-90K-T PSNR	Vimeo-90K-T SSIM	Vid4 PSNR	Vid4 SSIM
Bicubic	-/-	-	26.14	0.7292	31.32	0.8684	23.78	0.6347
RCAN [55]	-/-	-	28.78	0.8200	35.35	0.9251	25.46	0.7395
SwinIR [23]	-/-	11.9	29.05	0.8269	35.67	0.9287	25.68	0.7491
TOFlow [48]	5/7	-	27.98	0.7990	33.08	0.9054	25.89	0.7651
DUF	7/7	5.8	28.63	0.8251	-	-	27.33	0.8319
PFNL	7/7	3.0	29.63	0.8502	36.14	0.9363	26.73	0.8029
RBPN	7/7	12.2	30.09	0.8590	37.07	0.9435	27.12	0.8180
EDVR [44]	5/7	20.6	31.09	0.8800	37.61	0.9489	27.35	0.8264
MuCAN [21]	5/7	-	30.88	0.8750	37.32	0.9465	-	-
VSR-T [3]	5/7	32.6	31.19	0.8815	37.71	0.9494	27.36	0.8258
PSRT-sliding	5/-	14.8	31.32	0.8834	-	-	-	-
VRT	6/-	30.7	31.60	0.8888	-	-	-	-
PSRT-recurrent	6/-	10.8	31.88	0.8964	-	-	-	-
BasicVSR [4]	15/14	6.3	31.42	0.8909	37.18	0.9450	27.24	0.8251
IconVSR [4]	15/14	8.7	31.67	0.8948	37.47	0.9476	27.39	0.8279
BasicVSR++ [6]	30/14	7.3	32.39	0.9069	37.79	0.9500	27.79	0.8400
VRT	16/7	35.6	32.19	0.9006	38.20	0.9530	27.93	0.8425
RVRT [24]	30/14	10.8	32.75	0.9113	38.15	0.9527	27.99	0.8462
PSRT-recurrent	16/14	13.4	32.72	0.9106	38.27	0.9536	28.07	0.8485

SypNet [36] to calculate the optical flow. For clips in the Vimeo-90K and Vid4 test set, we measure the motion of the 4th and the 7th frames. For clips in the REDS test set, we measure the motion of the 3rd and the 5th frames. This arrangement is related to the common usage of sliding-window-based VSR models on these datasets: we use seven frames as input on Vimeo-90K, while we only use five frames on REDS. The optical flow result contains two maps, which are the movement in the x-direction $\mathcal{W}^x \in \mathbb{R}^{H\times W}$ and the movement in the y-direction $\mathcal{W}^y \in \mathbb{R}^{H\times W}$. We use the magnitude to indicate the movement of each pixel

$$\mathcal{W}^m_{i,j} = \sqrt{|\mathcal{W}^x_{i,j}|^2 + |\mathcal{W}^y_{i,j}|^2}, \quad \mathcal{W}^m \in \mathbb{R}^{H\times W}.$$

Figure 3 shows the variation curve of the total variation of the fine-tuned optical flow during training. The total variation is often used to indicate how smooth the optical flow is. The total variation of noise-contaminated optical flow is significantly larger than that of the noise-free optical flow. Given the optical flow $\{\mathcal{W}^x, \mathcal{W}^y\}$, the total variation is calculated as

$$\text{total variation} = \frac{1}{2HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(|\mathcal{W}^x_{i,j-1} - \mathcal{W}^x_{i,j}| + |\mathcal{W}^y_{i+1,j} - \mathcal{W}^y_{i,j}|).$$

We calculate the total variation every 5000 iterations.

Figure 4 shows the performance differences between VSR Transformers with and without alignment modules for different pixel movements. The VSR Transformer backbone used in this figure contains 16 Multi-Frame Self-Attention Blocks (MFSABs). Similar to SwinIR [23], we add shortcut

connections every 4 MFSABs. The feature dimension is 120, and the head number of the multi-head self-attention is 6. To plot the differences, we first partition the pixels into different groups according to their movement conditions and then calculate the mean square error for each group. We subtract the mean square errors of the VSR Transformer with alignment from the errors of the VSR Transformer without alignment. Thus, the parts greater than zero indicate better performance without alignment.

For the first sub-figure, we study the image alignment. The window size is set to 8. We keep the other settings for the second sub-figure and enlarge the window size to 12. For the third sub-figure, we replace the image alignment to feature alignment. In addition to the 2D convolution feature extraction, we add one CNN residual block to extract deep features. These experiments are performed under the same training settings described in Section 3 in the main text.

Table 1 shows the quantitative comparison of different VSR methods. For VSR CNNs, we use ten residual blocks [26] to extract features for all the input frames. We concatenate the features and reduce the channel number using a convolution layer. Five residual blocks are then used to conduct further processing. The VSR Transformers involved in this table share similar backbone architecture with the VSR Transformers in Figure 4, which contains 16 MFSABs with shortcut connections every 4 MFSABs. The feature dimension is 120, and the head number of the multi-head self-attention is 6. The training method is the same as described in section 3 of the main text. For the methods in which the flow network is not fixed, the learning rate for the flow network is 2.5×10^{-5}. For the first 5,000 iterations, the flow network is fixed.

Figure 5 shows the curves of some of the methods in Table 1. One can refer to Table 1 for the experimental details. The calculation of the movement distribution is the same as in Figure 2. The only difference is that we show the percentage in Figure 5, not the counts of pixels.

Table 2 and Table 3 share the same training setting and Transformer backbone. The VSR Transformer backbone contains 36 MFSABs with shortcut connections every 6 MFSABs. The window size is set to 8. The channel size of the transformer and head size are set to 144 and 6. The training method is the same as described in Section 3 of the main text. For the methods in which the flow network is not fixed, the learning rate for the flow net is 5×10^{-5}. For the first 20,000 iterations, the flow network is fixed. The implementation of different alignment methods is the same as described in Section 3 of the main text.

Table 4 shows the quantitative comparison between the proposed method and the state-of-the-art VSR methods. Due to the space limit, we only show limited results in the main text; the full version is shown in Table 8.

The architecture of the PSRT-recurrent is shown in Figure 12 and described in Section A. For each feature propagation block in PSRT-recurrent, we use 18 MFSABs with shortcut connections every 6 MFSABs. The feature size is set to 120, and the number of attention heads is 6.

The PSRT-sliding backbone contains 36 MFSABs with shortcut connections every 6 MFSABs. The window size is set to 8. The channel size of the transformer and head size are set to 144 and 6.

For the PSRT-recurrent with 16 input frames and the PSRT-sliding method, the total training iteration is 600K. The initial learning rate for these experiments is set to 2×10^{-4}. All other settings remain unchanged. For the PSRT-recurrent trained on Vimeo-90K, we follow [6] to initialize the model with the well-trained model using REDS. We fine-tune it for the other 300K iterations. The initial learning rate is 1×10^{-4}.

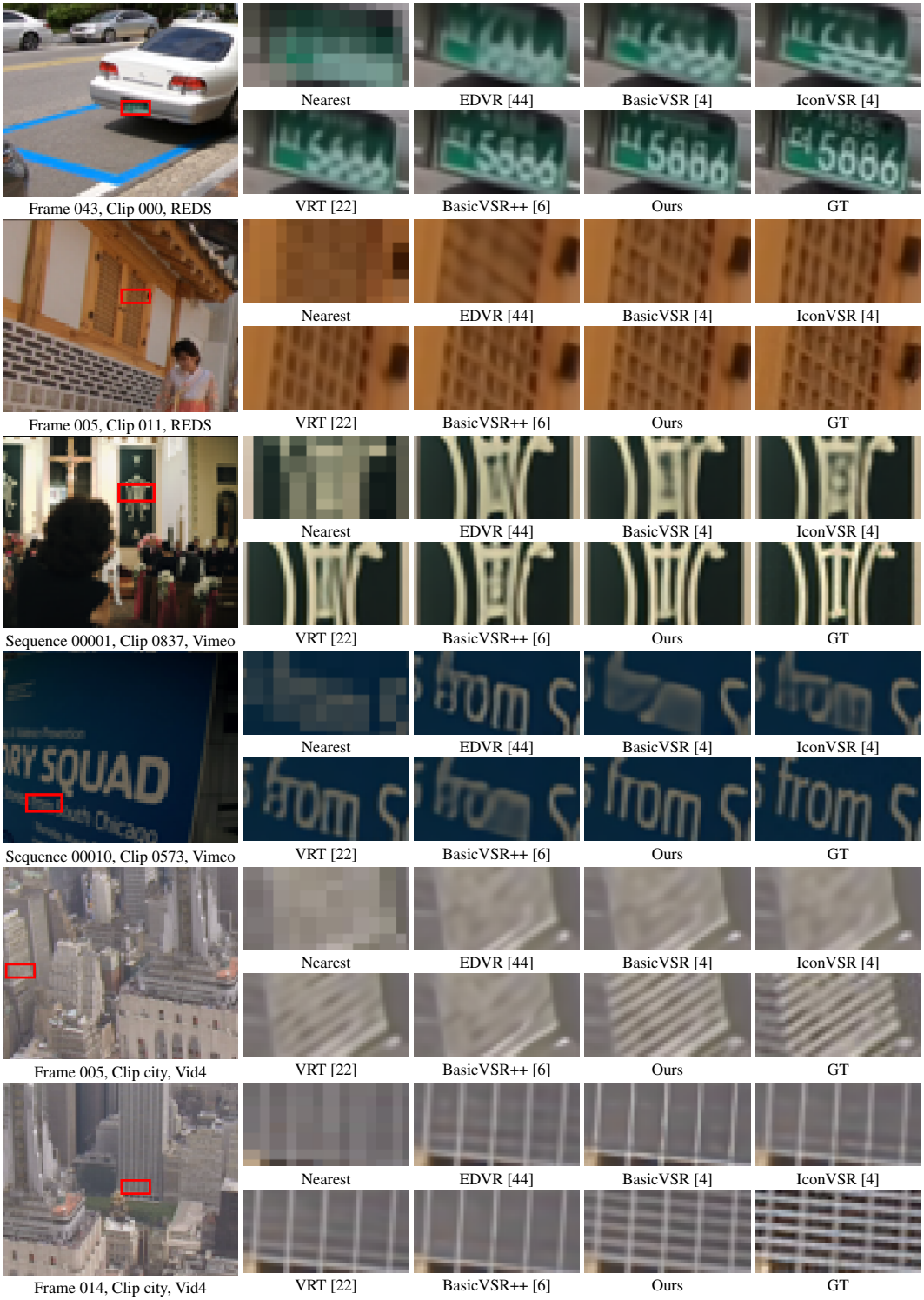

Figure 15: Visual comparison of VSR ($\times 4$) on REDS, Vimeo-90K and Vid4 datasets.