# Video Frame Interpolation Transformer

Zhihao Shi[*1] Xiangyu Xu[*†2] Xiaohong Liu[3] Jun Chen[1] Ming-Hsuan Yang[4,5,6]

[1]McMaster Univeristy [2]Nanyang Technological University [3]Shanghai Jiao Tong University
[4]University of California, Merced [5]Yonsei University [6]Google Research

## Abstract

*Existing methods for video interpolation heavily rely on deep convolution neural networks, and thus suffer from their intrinsic limitations, such as content-agnostic kernel weights and restricted receptive field. To address these issues, we propose a Transformer-based video interpolation framework that allows content-aware aggregation weights and considers long-range dependencies with the self-attention operations. To avoid the high computational cost of global self-attention, we introduce the concept of local attention into video interpolation and extend it to the spatial-temporal domain. Furthermore, we propose a space-time separation strategy to save memory usage, which also improves performance. In addition, we develop a multi-scale frame synthesis scheme to fully realize the potential of Transformers. Extensive experiments demonstrate the proposed model performs favorably against the state-of-the-art methods both quantitatively and qualitatively on a variety of benchmark datasets. The code and models are released at* https://github.com/zhshi0816/Video-Frame-Interpolation-Transformer.

## 1. Introduction

Video frame interpolation aims to temporally upsample an input video by synthesizing new frames between existing ones. It is a fundamental problem in computer vision that involves the understanding of motions, structures, and natural image distributions, which facilitates numerous downstream applications, such as image restoration [5, 52], virtual reality [1], and medical imaging [22].

Most of the state-of-the-art video frame interpolation methods are based on deep convolution neural networks (CNNs) [3, 20, 25, 29, 30, 32, 37, 53]. While achieving the state-of-the-art performance, these CNN-based architectures usually suffer from two major drawbacks. First, the convolution layer is content-agnostic, where the same kernels are used to convolve with different locations of different inputs. While this design can serve as a desirable inductive bias for image recognition models to acquire translational equivalence [24], it is not always suitable for video interpolation which involves a complex motion-compensation process that is spatially-variant and content-dependent. Thus, adopting CNN backbones may restrict the ability of adaptive motion modeling and potentially limits further development of video interpolation models.

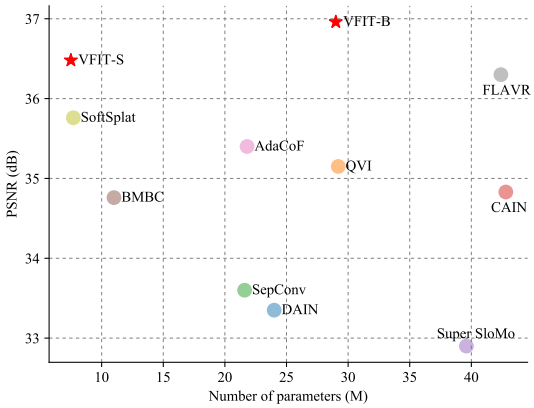



Figure 1. Comparison of performance and model size using the Vimeo-90K dataset [54]. VFIT outperforms the state-of-the-art methods with fewer parameters. VFIT-S and VFIT-B denote the proposed small and base models.

Second, capturing long-range dependencies is of central importance in video interpolation for which large motion fields pose the most prominent challenges. However, most CNNs [25, 53] usually employ small convolution kernels (typically 3×3 as suggested by VGG [39]), which is inefficient in exploiting long-range information and thus less effective in synthesizing high-quality video frames. While it seems an easy fix to use larger kernels in the convolution layer, it significantly increases the number of model parameters and computational cost, thereby leading to poor local minimums in training without proper regularizations.

[*]These authors contributed equally.
[†]Corresponding author.

Moreover, simply stacking multiple small kernel layers for a larger receptive field does not fully resolve this problem either, as distant dependencies cannot be effectively learned in a multi-hop fashion [45].

On the other hand, Transformers [43], which are initially designed for natural language processing (NLP) to efficiently model long-range dependencies between input and output, naturally overcome the above drawbacks of CNN-based algorithms, and are in particular suitable for the task of video interpolation. Motivated by the success in NLP, several methods recently adapt Transformers to computer vision and demonstrate promising results on various tasks, such as image classification [13, 41], semantic segmentation [44], object detection [8], and 3D reconstruction [51]. Nevertheless, how to effectively apply Transformers to video interpolation that involves an extra temporal dimension remains an open yet challenging problem.

In this work, we propose the Video Frame Interpolation Transformer (VFIT) for effective video interpolation. Compared with typical Transformers [8, 9, 13] where the basic modules are largely borrowed from the original NLP model [43], there are three distinguished designs in the proposed VFIT to generate photo-realistic and temporally-coherent frames. First, the original Transformer [43] is based on a self-attention layer that interacts with the input elements (e.g., pixels) globally. As this global operation has quadratic complexity with regard to the number of elements, directly applying it to our task leads to extremely high memory and computational cost due to the high-dimensionality nature of videos. Several methods [7, 9] circumvent this problem by dividing the feature maps into patches and treating each patch as a new element in the self-attention. However, this strategy cannot model fine-grained dependencies between pixels inside each patch which are important for synthesizing realistic details. Moreover, it may introduce edge artifacts around patch borders. In contrast, we introduce the local attention mechanism of Swin [27] into VFIT to address the complexity issue while retaining the capability of modeling long-range dependencies with its shift-window scheme. We demonstrate that with proper development and adaptation, the local attention mechanism originally used for image recognition can effectively improve the video interpolation performance with a smaller amount of parameters as shown in Figure 1.

Second, the original local attention mechanism [27] is only suitable for image input and cannot be easily used for the video interpolation task where an extra temporal dimension is involved. To address this issue, we generalize the concept of local attention to spatial-temporal domain, which leads to the Spatial-Temporal Swin attention layer (STS) that is compatible with videos. However, this simple extension could lead to memory issues when using large window sizes. To make our model more memory efficient, we further devise a space-time separable version of STS, called Sep-STS, by factorizing the spatial-temporal self-attention. Interestingly, Sep-STS not only effectively reduces memory usage but also considerably improves video interpolation performance.

To exploit the full potential of our Sep-STS, we propose a new multi-scale kernel-prediction framework which can better handle multi-scale motion and structures in diverse videos, and generates high-quality video interpolation results in a coarse-to-fine manner. The proposed VFIT is concise, flexible, light-weight, high-performing, fast, and memory-efficient. As shown in Figure 1, a small model (VFIT-S) already outperforms the state-of-the-art FLAVR method [21] by 0.18 dB with only **17.7%** of its parameters, while our base model (VFIT-B) achieves **0.66 dB** improvement with 68.4% of its parameters.

## 2. Related Work

**Video frame interpolation.** Existing video frame interpolation methods can be broadly classified into three categories: flow-based [3, 20, 32, 38, 53], kernel-based [25, 29–31], and direct-regression-based methods [22].

The flow-based methods [3, 20, 32, 53] generate intermediate frames by warping pixels from the source images according to predicted optical flow. Although these methods perform well, they are usually based on simplified motion assumptions such as linear [20] and quadratic [53], limiting their performance in many real-world scenarios where the assumptions are violated.

Unlike the flow-based approaches, the kernel-based methods [25, 29–31] do not rely on any prescribed assumptions and thus generalize better to diverse videos. For example, SepConv [30] predicts adaptive separable kernels to aggregate source pixels of the input, and AdaCoF [25] learns deformable spatially-variant kernels that are used to convolve with the input frames to produce the target frame. However, these approaches usually apply the kernel prediction modules at one scale and thereby cannot effectively handle complex motions and structures that could appear in different scales. Moreover, these CNN-based methods do not account for long-range dependency among pixels. In contrast, we propose a multi-scale Transformer-based kernel prediction module, which achieves higher-quality results for video interpolation as will be shown in Section 4.

Recently, Kalluri *et al*. [21] propose a CNN model to directly regress the target frame, which achieves the state-of-the-art results. As shown in Figure 1, the proposed VFIT outperforms this method by a clear margin with fewer parameters, which clearly shows the advantages of Transformers in video interpolation.

**Vision Transformer.** Transformers have recently been applied to various vision tasks, such as visual classification [4,

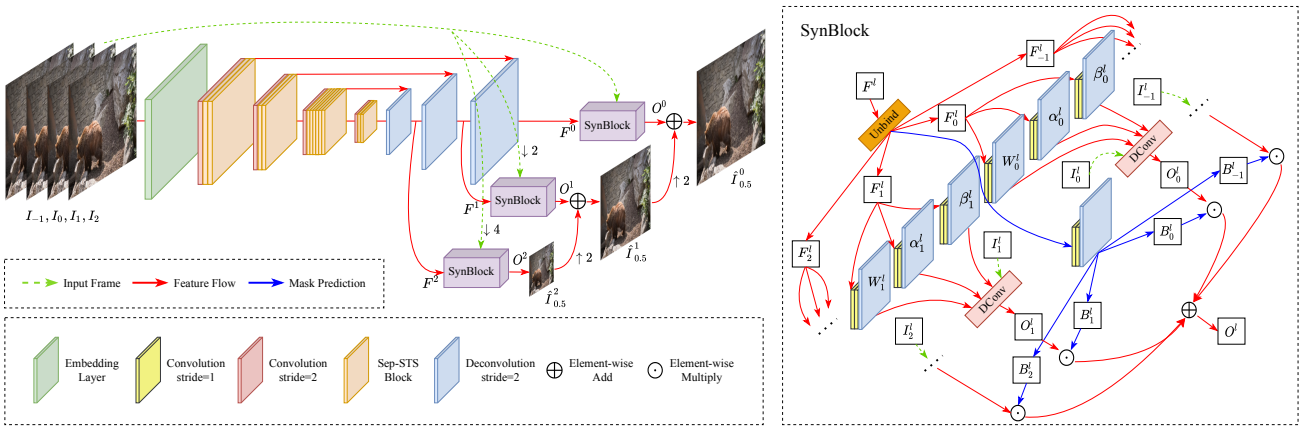



Figure 2. Overview of the proposed VFIT. We first use an embedding layer to transform input frames into shallow features, followed by a Transformer-based encoder-decoder network to extract deep hierarchical features. These features together with the input frames are fed to a multi-scale frame synthesis network that is composed of three SynBlocks to obtain the final output. "$\downarrow n$" and "$\uparrow n$" denote downsampling and upsampling by a factor of $n$, respectively. "DConv" represents the generalized deformable convolution in [50]. Note the SynBlock can be seen as a multi-frame extension of AdaCoF [25] originating from STPAN [50]. Please find more detailed explanations in Section 3.

13, 27, 42, 48], object detection [8], semantic segmentation [44], 3D reconstruction [51], and image restoration [9]. Nevertheless, it has not been exploited for video frame interpolation. In this work, we propose VFIT that achieves the state-of-the-art performance with a light-weight model. To overcome the high computational cost caused by global self-attention, we introduce the local attention mechanism of Swin [27] to avoid the complexity issues while retaining the ability of long-range dependency modeling. We note that one concurrent work [26] also uses the local attention for low-level vision tasks. However, it only considers image input and cannot deal with videos which are more challenging to handle due to the extra temporal dimension. In contrast, we extend the concept of local attention to the spatial-temporal domain to enable Transformer-based video interpolation, and propose a space-time separation strategy which not only saves memory usage but also acts as an effective regularization for performance gains.

## 3. Proposed Method

Figure 2 shows an overview of the proposed model. Similar to existing methods [21, 29, 30, 53], to synthesize an intermediate frame $I_{0.5}$, we use its $T$ neighboring frames $I_{\{-(\lfloor \frac{T}{2} \rfloor -1),\cdots,0,1,\cdots,\lceil \frac{T}{2} \rceil\}}$ as the input. Specifically, the input frames are $I_{-1}, I_0, I_1, I_2$ when $T$ is 4.

The proposed VFIT consists of three modules: shallow feature embedding, deep feature learning, and final frame synthesis. First, the embedding layer takes the input frames and generates shallow features for the deep feature learning module. Similar to [27], the shallow embedding is realized with a convolution layer, where we adopt the 3D convolution rather than its 2D counterpart in [27] to better encode the spatial-temporal features of the input sequence. Next, we feed the shallow features to the deep module to extract hierarchical feature representations $\{F^l, l = 0, 1, 2\}$ to capture the multi-scale motion information (Section 3.1). Finally, an intermediate frame $\hat{I}_{0.5}$ can be generated by the frame synthesis blocks (SynBlocks in Figure 2) using the deep features $F^l$ (Section 3.2).

### 3.1. Learning Deep Features

As shown in Figure 2, we use a Transformer-based encoder-decoder architecture for learning features. The encoder is composed of four stages, where each stage starts with a 3D convolution layer using a stride of 2 to downsample the input features, and the downsampling layer is followed by several Sep-STS blocks which are the main components of our framework. For the decoder, we use a light-weight structure that only has three 3D deconvolution layers with a stride of 2 to upsample the low-resolution feature maps. Note that we only resize the spatial dimension of the features throughout our network and leave the temporal size unchanged. Next, we provide more explanations about the proposed Sep-STS block.

**Local attention.** Existing Transformers [8, 13, 43] mainly adopt a global attention mechanism to aggregate information from the input, which could cause extremely high memory and computational cost for video frame interpolation. A straightforward solution to this problem is to directly divide the feature maps into patches and treat each patch as a new element in the global attention [7, 9]. This strategy is equivalent to aggressively downsampling the input with pixel shuffle [36] (downsampling factor equals to the patch size) and cannot well reconstruct high-quality image details which require fine-grained dependency modeling between pixels.

In this work, we introduce the local attention mechanism of Swin Transformer [27], which can effectively ad-

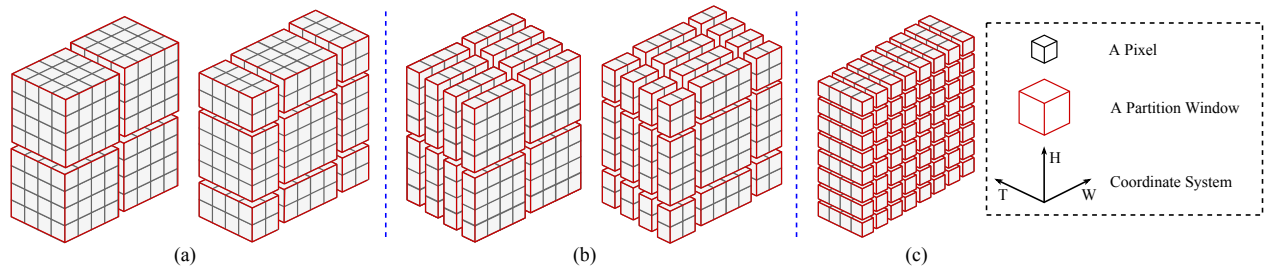



Figure 3. Illustration of different local partition strategies. (a) Regular and shifted partitions for spatial-temporal cubes of STS. (b) Regular and shifted partitions for spatial windows of Sep-STS. (c) Temporal vector partition for Sep-STS.

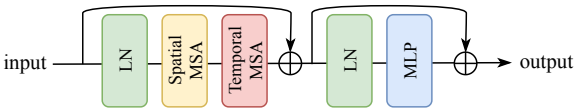



Figure 4. Illustration of the Sep-STS block. The Spatial MSA and Temporal MSA represent the multi-head self-attentions in spatial and temporal local windows, respectively (Section 3.1).

dress the above issues. First, as the self-attention of Swin is computed inside local windows, it naturally avoids the heavy computational burden of global attentions. Second, Swin employs a shifted-window partition strategy to connect different local regions, and alternatively using regular and shifted-window partitions enables long-range dependency modeling. Nevertheless, this method is designed for image applications and cannot be easily applied to videos.

**Spatial-temporal local attention.** To make the Swin Transformer compatible with video inputs, we generalize the local attention mechanism to spatial-temporal space and propose the STS attention. As shown in Figure 3(a), the STS is conceptually similar to Swin but involves an extra temporal dimension.

Given an input feature of size $C \times T \times H \times W$ where $C$, $T$, $H$, $W$ respectively represent the channel, time, height, and width dimensions, we first partition it to $\frac{HW}{M^2}$ non-overlapped 3D sub-cubes with the shape of each cube as $T \times M \times M$ (Figure 3(a)-left) and then perform standard multi-head self-attention (MSA) on each sub-cube. Note that each element of this cube is a $C$-dimensional feature vector, and we omit the channel dimension when describing the partition strategies for simplicity. Once all the sub-cubes are processed, we merge them back to recover the original shape of the input. In order to bridge connections across neighboring cubes, we adopt a shifted-cube partition strategy, which shifts the cubes to top-left by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ pixels (Figure 3 (a-right).

**Separation of space and time.** Although the above STS can handle video inputs, it may suffer from memory issues when dealing with large cube sizes, *i.e.*, large $T$ or $M$. To alleviate this issue, we propose the Sep-STS by separating the spatial-temporal computations into space and time.

First, for the computation in space, given an input feature map with a size of $C \times T \times H \times W$, we first partition it into $\frac{THW}{M^2}$ non-overlapped 2D sub-windows with a size of $M \times M$ as shown in Figure 3(b)-left, and then perform the standard MSA for each sub-window. For connecting different windows, as we restrict our computations in 2D here, we simply use the shifted window partition strategy of the Swin for each frame as shown in Figure 3(b)-right.

Second, for the computation in the temporal dimension, we reshape the input feature map into $HW$ temporal vectors with a length of $T$ as shown in Figure 3(c) and perform MSA inside each vector such that the dependencies across frames can be modeled. This step complements the self-attention in the spatial domain, and thus the two operations need to be used together to process videos.

**Sep-STS block.** Based on the Sep-STS attention, we devise our main component, the Sep-STS block, which is composed of separated spatial and temporal attention modules as well as an MLP (Figure 4). The MLP adopts a two-layer structure and uses the GELU function [17] for activation. Similar to [27], we apply Layer Normalization (LN) [2] and residual connections [16] in this block to stabilize training. Similar to Swin, the regular and shifted partitions are employed alternatively for consecutive Sep-STS blocks to model long-range spatial-temporal dependencies.

**Memory usage.** The Sep-STS attention factorizes a computationally expensive operation into two lighter operations in space and time, which effectively lessens the memory usage reducing from $\mathcal{O}((TMM) \cdot THW)$ of the STS to $\mathcal{O}((T + MM) \cdot THW)$ of our Sep-STS.

During training, compared to the STS baseline, we observe a 26.2% GPU memory reduction by using our Sep-STS. As the window size $MM$ is usually much larger than the number of input frames $T$, this reduction ratio essentially relies on $T$ which we set as 4 by default following the settings of the state-of-the-art algorithms [11, 21, 53]. Since the proposed framework is flexible and can be used for arbitrary number of frames, the Sep-STS can potentially give more significant memory reduction for a larger $T$. In addition, the space-time separation strategy can also reduce the computational cost similar to the memory usage. However, as the Sep-STS is naively implemented with two separate

PyTorch [33] layers in our experiments, its run-time is in fact similar to that of the STS. Potentially, optimizing its implementation with a customized CUDA kernel may further improve the efficiency.

**Discussions.** In this work, we explore the concept of local attention for Transformer-based video interpolation. Similar concepts have been adopted in other recent methods, such as the local relation network [19], stand-alone network [35], and Swin [27]. Nevertheless, these algorithms are designed for images, and less attention is paid to exploiting local attention mechanisms for videos due to difficulties caused by the extra temporal dimension. In addition, existing methods mainly focus on image recognition tasks that are generally seen as high-level vision tasks, while in this work we emphasize more on motion modeling and appearance reconstruction. In this work, we focus on the temporal extension of local attention modules for effective video frame interpolation. We explore the space-time separable local attention, which is in spirit similar to MobileNet [18] that improves a standard convolution by factorizing it into a depthwise convolution and a pointwise convolution. Furthermore, we propose a multi-scale kernel-prediction framework to fully exploit the features learned by local attention, as introduced next.

### 3.2. Frame Synthesis

With the features from the proposed encoder-decoder network, our VFIT synthesizes the output image by predicting spatially-variant kernels to adaptively fuse the source frames. Different from existing kernel-based video interpolation methods [25, 29, 30, 37], we propose a multi-scale kernel-prediction framework using the hierarchical feature $\{F^l, l = 0, 1, 2\}$ as shown in Figure 2.

The frame synthesis network of VFIT is composed of three SynBlocks that make predictions at different scales, and each SynBlock is a kernel prediction network. VFIT fuses these multi-scale predictions to generate the final result by:

$$\hat{I}^l_{0.5} = f_{\text{up}}(\hat{I}^{l+1}_{0.5}) + O^l, \tag{1}$$

$$O^l = f^l_{\text{syn}}(F^l, I^l_{\{-(\lfloor \frac{T}{2} \rfloor - 1), \cdots, \lceil \frac{T}{2} \rceil\}}), \tag{2}$$

where $l = 0, 1, 2$ represent different scales from fine to coarse, and $f_{\text{up}}$ denotes the bilinear upsampling function. The synthesized frame at a finer scale $\hat{I}^l_{0.5}$ can be obtained by merging the upsampled output from the coarse scale ($f_{\text{up}}(\hat{I}^{l+1}_{0.5})$) and the prediction of the current SynBlock ($O^l$). The output at the finest scale $\hat{I}^0_{0.5}$ is the final result of our VFIT, *i.e.*, $\hat{I}_{0.5} = \hat{I}^0_{0.5}$, and the initial value $\hat{I}^3_{0.5} = 0$. Here, $f^l_{\text{syn}}$ is the $l$-th SynBlock which takes the spatial-temporal feature $F^l$ and the frame sequence $I^l_{\{-(\lfloor \frac{T}{2} \rfloor - 1), \cdots, \lceil \frac{T}{2} \rceil\}}$ as input, and $I^l_t$ represents a frame $I_t$ downsampled by a factor of $2^l$ with bilinear interpolation, where $I^0_t$ is equivalent to the original frame without downsampling.

**SynBlock.** Given the input feature map $F^l \in \mathbb{R}^{C \times T \times H \times W}$, the SynBlock generates its prediction at the $l$-th scale by estimating a set of generalized deformable kernels [50] to aggregate the information from the source frames.

As illustrated in Figure 2, we first unbind $F^l$ at the temporal dimension to get $T$ separate feature maps for all input frames, denoted as $F^l_{\{-(\lfloor \frac{T}{2} \rfloor - 1), \cdots, \lceil \frac{T}{2} \rceil\}}$, and for each frame $t$, $F^l_t \in \mathbb{R}^{C \times H \times W}$. Then we feed $F^l_t$ into three small 2D CNNs to obtain the per-pixel deformable kernels for frame $I^l_t$, including the kernel weights $W^l_t \in \mathbb{R}^{K \times H \times W}$, horizontal offsets $\alpha^l_t \in \mathbb{R}^{K \times H \times W}$, and vertical offsets $\beta^l_t \in \mathbb{R}^{K \times H \times W}$, where $K$ is the number of sampling locations of each kernel.

With the predicted kernels we obtain the output of the SynBlock at location $(x, y)$ for frame $t$ as:

$$O^l_t(x, y) = \sum_{k=1}^{K} W^l_t(k, x, y) I^l_t(x + \alpha^l_t(k, x, y), y + \beta^l_t(k, x, y)),$$

which aggregates neighboring pixels around $(x, y)$ with adaptive weights $W$ similar to [50].

Finally, we generate the output at scale $l$ by blending $O^l_t$ of all frames with learned masks. Specifically, we concatenate the feature maps $F^l_{\{-(\lfloor \frac{T}{2} \rfloor - 1), \cdots, \lceil \frac{T}{2} \rceil\}}$ at the channel dimension and send the concatenated features to a small 2D CNN to produce $T$ blending masks $B^l_{\{-(\lfloor \frac{T}{2} \rfloor - 1), \cdots, \lceil \frac{T}{2} \rceil\}}$. Note that we use a softmax function as the last layer of the CNN to normalize the masks along the temporal dimension. The final output of the SynBlock $f^l_{\text{syn}}$ is generated by:

$$O^l = \sum_t B^l_t \cdot O^l_t. \tag{3}$$

Note that this SynBlock can be seen as a multi-frame extension of [25, 37] that originate from the generalized deformable kernels of STPAN [49].

## 4. Experiments

### 4.1. Implementation Details

**Network.** As shown in Figure 2, the VFIT encoder consists of four stages that have 2, 2, 6, and 2 Sep-STS blocks, respectively. The skip connections between the encoder and decoder are realized with concatenation. For all three SynBlocks, we set the deformable kernel size as $K = 5 \times 5$. We present two variants of VFIT: a base model VFIT-B and a small one VFIT-S, where the model size of VFIT-S is about 25% of VFIT-B. The two models use the same architecture, and the only difference is the channel dimension of each stage, where we shrink the channels by half for VFIT-S.

Table 1. Quantitative comparisons on the Vimeo-90K, UCF101, and DAVIS datasets. Numbers in bold indicate the best performance and underscored numbers indicate the second best.

| Method | # Parameters (M) | Vimeo-90K | | UCF101 | | DAVIS | |
|---|---|---|---|---|---|---|---|
| | | PSNR (↑) | SSIM (↑) | PSNR (↑) | SSIM (↑) | PSNR (↑) | SSIM (↑) |
| SuperSloMo [20] | 39.6 | 32.90 | 0.957 | 32.33 | 0.960 | 25.65 | 0.857 |
| DAIN [3] | 24.0 | 33.35 | 0.945 | 31.64 | 0.957 | 26.12 | 0.870 |
| SepConv [30] | 21.6 | 33.60 | 0.944 | 31.97 | 0.943 | 26.21 | 0.857 |
| BMBC [32] | 11.0 | 34.76 | 0.965 | 32.61 | 0.955 | 26.42 | 0.868 |
| CAIN [12] | 42.8 | 34.83 | 0.970 | 32.52 | 0.968 | 27.21 | 0.873 |
| AdaCoF [25] | 21.8 | 35.40 | 0.971 | 32.71 | 0.969 | 26.49 | 0.866 |
| QVI [53] | 29.2 | 35.15 | 0.971 | 32.89 | 0.970 | 27.17 | 0.874 |
| SoftSplat [28] | 7.7 | 35.76 | 0.972 | 32.89 | 0.970 | 27.42 | 0.878 |
| FLAVR [21] | 42.4 | 36.30 | 0.975 | 33.33 | **0.971** | 27.44 | 0.874 |
| VFIT-S | **7.5** | 36.48 | 0.976 | 33.36 | **0.971** | 27.92 | 0.885 |
| VFIT-B | 29.0 | **36.96** | **0.978** | **33.44** | **0.971** | **28.09** | **0.888** |

**Training.** For training our network, we employ a simple $\ell_1$ loss: $||I_{0.5} - \hat{I}_{0.5}||$, where $I_{0.5}$ is the ground truth. We use the AdaMax optimizer [23] with $\beta_1 = 0.9, \beta_2 = 0.999$. The training batch size is set as 4. We train the models for 100 epochs, where the learning rate is initially set as $2e^{-4}$ and gradually decayed to $1e^{-6}$.

**Dataset.** Similar to [21], we adopt the Vimeo-90K septuplet training set [54] to learn our models, which consists of 64612 seven-frame sequences with a resolution of $448 \times 256$. The first, third, fifth, and seventh frames of each sequence correspond to $I_{-1}, I_0, I_1, I_2$ in Figure 2 and are used to predict the fourth frame corresponding to $I_{0.5}$. For data augmentation, we randomly crop $256 \times 256$ image patches from frames, and perform horizontal and vertical flipping, as well as temporal order reverse.

We evaluate the models on the widely-used benchmark datasets, including the Vimeo-90K septuplet test set [54], UCF101 dataset [40], and DAVIS dataset [34]. Following [21, 53], we report performance on 100 quintuples generated from UCF101 and 2847 quintuples from DAVIS.

## 4.2. Evaluation against the State of the Arts

We evaluate the proposed algorithm against the state-of-the-art video interpolation methods: SepConv [30], DAIN [3], SuperSloMo [20], CAIN [12], BMBC [32], AdaCoF [25], SoftSplat [28], QVI [53], and FLAVR [21]. Among these methods, SuperSloMo, DAIN, CAIN, QVI, AdaCoF, and FLAVR are trained on the same training data as our models. For SepConv and BMBC, as the training code is not available, we directly use the pre-trained models for evaluation. The results of SoftSplat [28] are kindly provided by the authors.

We show quantitative evaluations in Table 1 where the PSNR and SSIM [46] are used for image quality assessment similar to previous works. Thanks to the learning capacity of the Sep-STS block, the proposed VFIT achieves better performance than the evaluated CNN-based methods, demonstrating the superiority of using Transformers for video interpolation. Specifically, with only 7.5M parameters, the VFIT-S is able to outperform FLAVR, the best video interpolation method to date, on all the evaluated datasets. Furthermore, the VFIT-B achieves more significant improvements over FLAVR (0.66 dB on Vimeo-90K and 0.65 dB on DAVIS). Since the videos of UCF101 have relatively low qualities with low image resolutions and slow motion as explained in [53], our performance gain is less significant. Note that the large improvement of the VFIT comes solely from the architecture design without relying on any external information, which differs sharply from several prior works [3, 28, 53] that use pre-trained optical flow and/or depth models and thus implicitly benefit from additional motion and/or depth labels.

Table 2. Run-time of evaluated methods in seconds per frame. The models are tested on a desktop with an Intel Core i7-8700K CPU and an NVIDIA GTX 2080 Ti GPU. The results are averaged on the Vimeo-90K dataset.

| Method | BMBC | QVI | FLAVR | VFIT-S | VFIT-B |
|---|---|---|---|---|---|
| Run-time | 0.57 | 0.08 | 0.15 | 0.08 | 0.14 |

In addition, we provide qualitative comparisons in Figure 5, where the proposed VFIT generates visually more pleasing results with clearer structures and fewer distortions than the baseline approaches. Moreover, to evaluate the accuracy of the interpolation results, we show overlaps of the interpolated frame and the corresponding ground truth in Figure 6. The overlapped images of VFIT are much clearer than the baselines, *i.e.* closer to the ground truth, indicating better capabilities of VFIT in motion modeling.

We also present the run-time of our method in Table 2. The run-time performance of the VFIT is on par with the best performing CNN-based algorithms, which facilitates its deployment in vision applications.

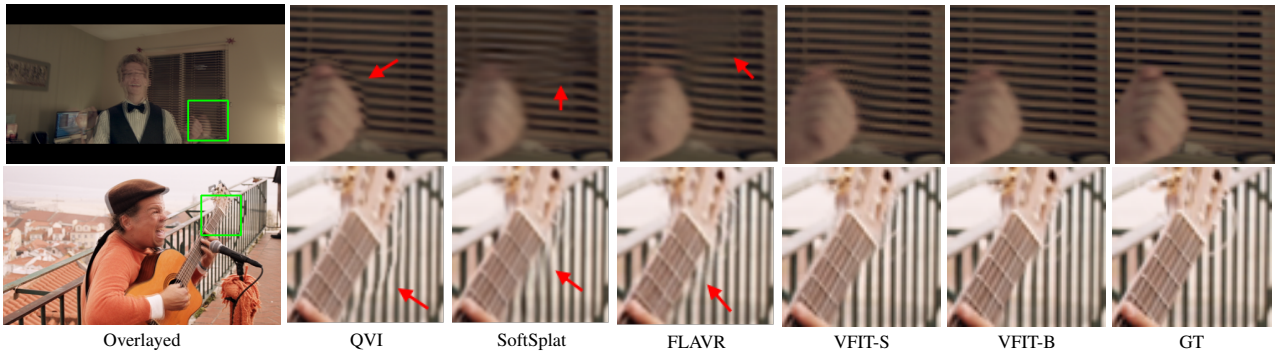



Figure 5. Qualitative comparisons against the state-of-the-art video interpolation algorithms. The VFIT generates higher-quality results with clearer structures and fewer distortions.


Figure 6. Overlap of interpolated frames and the corresponding ground truth, where a clearer overlapped image indicates a more accurate prediction. Note that for the second example, as the predictions of the baseline methods and the ground truth are not well aligned, the overlap of the red and white regions presents a blurry pink color.

### 4.3. Ablation Study

We conduct the ablation studies on the Vimeo-90K dataset. As we notice that the training process converges quickly in the early training stage where differences between models can already be distinguished, we train all models in this study for 20 epochs to accelerate the development and concentrate on the most essential parts of VFIT.

**Local attention.** In contrast to our model which introduces the local attention mechanism, several recent methods [7, 9] follow the basic structure of conventional Transformers in NLP to use global attention for vision applications, where the high computational cost of the global attention is circumvented by dividing input into patches and redefining each patch as a new element in self-attention. In our experiments, we also try this strategy by replacing each Sep-STS block of VFIT-B with a patch-based global-attention block, which is called VFIT-Global. As shown in Table 3, the result of VFIT-Global is lower than VFIT-B by as large as 0.84 dB, which emphasizes the essential role of local attention in Transformer-based video frame interpolation.

**Sep-STS.** To further validate the effectiveness of the Sep-STS block, we compare our VFIT-B with its two variants: 1) VFIT-CNN where all the Sep-STS blocks are replaced by convolutional ResBlocks [16], and each ResBlock is composed of two 3D convolution layers; and 2) VFIT-STS where the Sep-STS block is replaced by its inseparable counterpart, *i.e.*, the STS block.

As shown in Table 3, while VFIT-CNN uses more than two times parameters of VFIT-STS, these two models achieve similar results, demonstrating the advantages of using Transformers for video interpolation. Further, our base model VFIT-B, which uses the proposed Sep-STS as the building block, obtains even better performance than the VFIT-STS. It should be emphasized that the performance gain is significant as the Sep-STS block is initially designed to reduce memory usage as discussed in Section 3.1. This can be attributed to that the self-attention of the large-size sub-cubes in STS is relatively difficult to learn, and the space-time separation in Sep-STS can serve as a low-rank regularization [6] to remedy this issue.

To better analyze the performance of our models, we further compare with the baselines under different motion conditions. Following [15, 47], we split the Vimeo-90K test set into fast, medium, and slow motions, respectively. Table 4 shows VFIT-B outperforms VFIT-CNN by 0.43 dB on fast motion, 0.16 dB on medium motion, and 0.10 dB on slow motion, highlighting the exceptional capability of the proposed Sep-STS in handling challenging large-motion scenarios. We also provide interpolated frames from a video

Table 3. Effectiveness of the proposed Sep-STS block.

| Method | PSNR | SSIM | #Parameters (M) |
|---|---|---|---|
| VFIT-B | **36.02** | **0.975** | 29.0 |
| VFIT-STS | 35.84 | 0.974 | 29.1 |
| VFIT-CNN | 35.82 | 0.973 | 65.4 |
| VFIT-Global | 35.18 | 0.971 | 42.4 |
| $M = 4$ | 35.82 | 0.974 | 29.0 |
| $M = 6$ | 35.90 | 0.974 | 29.0 |
| $M = 8$ | **36.02** | **0.975** | 29.0 |
| $M = 10$ | 35.93 | 0.974 | 29.0 |

Table 4. Comparison with the base models under different motion conditions.

| Method | Fast | Medium | Slow |
|---|---|---|---|
| VFIT-B | **33.23/0.954** | **35.91/0.976** | **38.36/0.987** |
| VFIT-STS | 32.91/0.950 | 35.77/0.975 | 38.27/0.987 |
| VFIT-CNN | 32.80/0.950 | 35.75/0.975 | 38.26/0.987 |
| VFIT-Global | 32.15/0.945 | 35.10/0.972 | 37.62/0.985 |

with fast motion in Figure 7 for comparisons.

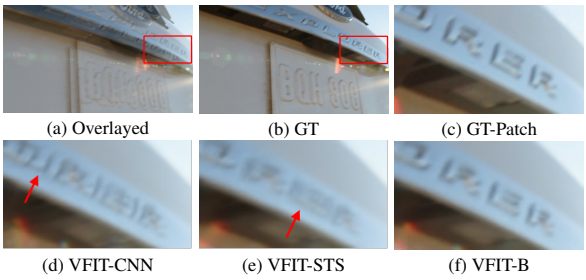
To analyze the effect of different window size of the Sep-STS, we evaluate the VFIT-B with $M = 4, 6, 8, 10$, respectively. Table 3 shows, our model performs better as the window size is increased until $M > 8$. Thus, we choose $M = 8$ as our default setting in this work.

**Multi-scale frame synthesis.** In Section 3.2, we propose a multi-scale kernel-prediction network for final frame synthesis. To verify the effectiveness of this design, we experiment with a single-scale variant of the VFIT, called VFIT-Single, by removing the second and third SynBlocks in Figure 2. This single-scale strategy is essentially similar to the ordinary kernel-prediction networks in [25, 29, 30]. The PSNR achieved by VFIT-Single is 35.54 dB, which is 0.48 dB lower than our base model VFIT-B. The large performance gap shows the importance of the multi-scale framework for fully realizing the potential of Transformers.

Note that we only apply the loss function to the final output, *i.e.*, the finest level output $\hat{I}^0_{0.5}$ of the multi-scale framework as introduced in Section 4.1. Alternatively, one may consider adding supervision to all-scale outputs of the network. However, we empirically find this scheme does not perform well.

**Resizing modules.** As illustrated in Figure 2, we use 3D convolution and deconvolution layers for downsampling and upsampling the feature maps. Motivated by the performance gain of our Sep-STS over CNN-based models, it is of great interest to explore the use of Transformer layers as the resizing modules for video frame interpolation.

To answer this question, we adopt the method in [14] which introduces a Transformer-based resizing module for video classification by downsampling the query of the self-attention layer. To enable Transformer-based upsampling, we extend the idea in [14] by upsampling the query with bilinear interpolation. We respectively replace the convolution and deconvolution layers of VFIT-B with these Transformer-based downsampling and upsampling modules, and refer to the two variants as VFIT-TD and VFIT-TU. As shown in Table 5, both VFIT-TD and VFIT-TU perform slightly worse than our base model with degraded run-time performance, indicating that the current designs of Transformer-based resizing operations in computer vision are less effective for complex motion modeling. This is a limitation of our current work, which will be an interesting problem for future research.



(a) Overlayed (b) GT (c) GT-Patch

(d) VFIT-CNN (e) VFIT-STS (f) VFIT-B

Figure 7. Interpolated frames from a video with fast motion. VFIT-CNN produces severe ghosting artifacts due to it incapability of handling large motion, while the result of VFIT-STS appears blurry. In contrast, the VFIT-B generates a higher-quality intermediate frame closer to the ground truth.

Table 5. Comparison with Transformer-based resizing modules.

| Method | PSNR | SSIM | Run-time (s) |
|---|---|---|---|
| VFIT-B | **36.02** | **0.975** | 0.14 |
| VFIT-TD | 35.92 | 0.974 | 0.17 |
| VFIT-TU | 35.97 | 0.974 | 0.20 |

## 5. Conclusion

In this paper, we propose a parameter, memory, and run-time efficient VFIT framework for video frame interpolation with the state-of-the-art performance. A significant part of our effort focuses on extending the local attention mechanism to the spatial-temporal space, and this module can be integrated in other video processing tasks. In addition, we demonstrate the effectiveness of a novel space-time separation scheme, which implies the necessity of well-structured regularizations in video Transformers. The architecture of VFIT is simple and compact, which can be effectively applied to numerous downstream vision tasks.

Similar to most existing kernel-based methods [25, 29, 30, 37], we only perform $2\times$ interpolation with VFIT. However, it can be easily extended to multi-frame interpolation by predicting kernels tied to different time steps or even arbitrary-time interpolation by taking time as an extra input similar to [10]. This will be part of our future work.

**Acknowledgement.** M.-H. Yang is supported in part by the NSF CAREER Grant #1149783.

## References


[1] Robert Anderson, David Gallup, Jonathan T Barron, Janne Kontkanen, Noah Snavely, Carlos Hernández, Sameer Agarwal, and Steven M Seitz. Jump: virtual reality video. *ACM Transactions on Graphics*, 35(6):1–13, 2016. 1

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[3] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 6

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning*, 2021. 2

[5] Tim Brooks and Jonathan T Barron. Learning to synthesize motion blur. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[6] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):1–37, 2011. 7

[7] Jiezhang Cao, Yawei Li, Kai Zhang, and Luc Van Gool. Video super-resolution transformer. *arXiv preprint arXiv:2106.06847*, 2021. 2, 3, 7

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision*, 2020. 2, 3

[9] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 2, 3, 7

[10] Xianhang Cheng and Zhenzhong Chen. Multiple video frame interpolation via enhanced deformable separable convolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 8

[11] Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu, Juwei Lu, Jin Tang, and Konstantinos N Plataniotis. All at once: Temporally adaptive multi-frame interpolation with advanced motion modeling. In *Proceedings of the European Conference on Computer Vision*, 2020. 4

[12] Myungsub Choi, Heewon Kim, Bohyung Han, Ning Xu, and Kyoung Mu Lee. Channel attention is all you need for video frame interpolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 6

[13] 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 *Proceedings of the International Conference on Learning Representations*, 2020. 2, 3

[14] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 8

[15] Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. Recurrent back-projection network for video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 7

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 4, 7

[17] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 4

[18] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 5

[19] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 5

[20] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 1, 2, 6

[21] Tarun Kalluri, Deepak Pathak, Manmohan Chandraker, and Du Tran. Flavr: Flow-agnostic video representations for fast frame interpolation. *arXiv preprint arXiv:2012.08512*, 2020. 2, 3, 4, 6

[22] Alexandros Karargyris and Nikolaos Bourbakis. Three-dimensional reconstruction of the digestive wall in capsule endoscopy videos using elastic video interpolation. *IEEE Transactions on Medical Imaging*, 30(4):957–971, 2010. 1, 2

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2014. 6

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 1

[25] Hyeongmin Lee, Taeoh Kim, Tae-young Chung, Daehyun Pak, Yuseok Ban, and Sangyoun Lee. Adacof: Adaptive collaboration of flows for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 2, 3, 5, 6, 8

[26] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte. Swinir: Image restoration using swin transformer. *arXiv preprint arXiv:2108.10257*, 2021. 3

[27] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2, 3, 4, 5

[28] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 6

[29] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 3, 5, 8

[30] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 1, 2, 3, 5, 6, 8

[31] Simon Niklaus, Long Mai, and Oliver Wang. Revisiting adaptive convolutions for video frame interpolation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2021. 2

[32] Junheum Park, Keunsoo Ko, Chul Lee, and Chang-Su Kim. Bmbc: Bilateral motion estimation with bilateral cost volume for video interpolation. In *Proceedings of the European Conference on Computer Vision*, 2020. 1, 2, 6

[33] 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 *Advances in Neural Information Processing Systems*, 2019. 5

[34] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 6

[35] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In *Advances in Neural Information Processing Systems*, 2019. 5

[36] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 3

[37] Zhihao Shi, Xiaohong Liu, Kangdi Shi, Linhui Dai, and Jun Chen. Video frame interpolation via generalized deformable convolution. *IEEE Transactions on Multimedia*, 2021. 1, 5, 8

[38] Hyeonjun Sim, Jihyong Oh, and Munchurl Kim. Xvfi: extreme video frame interpolation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2

[39] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations*, 2015. 1

[40] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 6

[41] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning*, 2021. 2

[42] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 2, 3

[44] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *Proceedings of the European Conference on Computer Vision*, 2020. 2, 3

[45] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[46] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6

[47] Xiaoyu Xiang, Yapeng Tian, Yulun Zhang, Yun Fu, Jan P Allebach, and Chenliang Xu. Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 7

[48] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision*, 2018. 2

[49] Gang Xu, Jun Xu, Zhen Li, Liang Wang, Xing Sun, and Ming-Ming Cheng. Temporal modulation network for controllable space-time video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6388–6397, 2021. 5

[50] Xiangyu Xu, Muchen Li, Wenxiu Sun, and Ming-Hsuan Yang. Learning spatial and spatio-temporal pixel aggregations for image and video denoising. *IEEE Transactions on Image Processing*, 2020. 3, 5

[51] Xiangyu Xu and Chen Change Loy. 3D human texture estimation from a single image with transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 2, 3

[52] Xiangyu Xu, Jinshan Pan, Yu-Jin Zhang, and Ming-Hsuan Yang. Motion blur kernel estimation via deep learning. *IEEE Transactions on Image Processing*, 27(1):194–205, 2017. 1

[53] Xiangyu Xu, Li Siyao, Wenxiu Sun, Qian Yin, and Ming-Hsuan Yang. Quadratic video interpolation. In *Advances in Neural Information Processing Systems*, 2019. 1, 2, 3, 4, 6

[54] 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. 1, 6