Deformable Video Transformer

Jue Wang[1] Lorenzo Torresani[1,2]
[1]Facebook AI Research [2]Dartmouth

Abstract

Video transformers have recently emerged as an effective alternative to convolutional networks for action classification. However, most prior video transformers adopt either global space-time attention or hand-defined strategies to compare patches within and across frames. These fixed attention schemes not only have high computational cost but, by comparing patches at predetermined locations, they neglect the motion dynamics in the video. In this paper, we introduce the Deformable Video Transformer (DVT), which dynamically predicts a small subset of video patches to attend for each query location based on motion information, thus allowing the model to decide where to look in the video based on correspondences across frames. Crucially, these motion-based correspondences are obtained at zero-cost from information stored in the compressed format of the video. Our deformable attention mechanism is optimized directly with respect to classification performance, thus eliminating the need for suboptimal hand-design of attention strategies. Experiments on four large-scale video benchmarks (Kinetics-400, Something-Something-V2, EPIC-KITCHENS and Diving-48) demonstrate that, compared to existing video transformers, our model achieves higher accuracy at the same or lower computational cost, and it attains state-of-the-art results on these four datasets.

1. Introduction

Although transformers [39] were originally proposed to address NLP problems, they have quickly gained popularity in computer vision after the introduction of the Vision Transformer (ViT) [9]. Compared to CNN architectures, which can model pixel dependencies only within the small receptive fields of convolutional filters, ViTs offer the benefit of capturing longer-range dependencies. This is achieved by means of the self-attention operation, which entails comparing features extracted from image patches at different locations. The downside of self-attention is that it causes a high computational cost if executed globally over all pairs of image patches. In fact, the complexity of global self-attention is $\mathcal{O}(S^2)$ where S is the number of patches, which can be in the thousands.

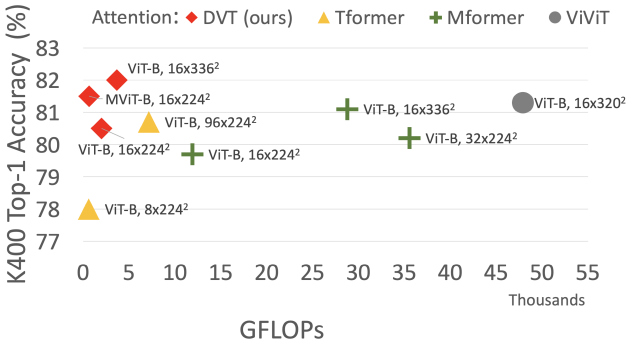

Figure 1. Computational cost vs classification accuracy on K400 for video transformers pretrained on ImageNet-21K. Red diamonds denote models employing our proposed attention scheme (DVT). Our method achieves consistently better accuracy and lower computational cost compared to prior attention strategies (Tformer [5], ViViT [2], Mformer [31]). Text tags next to markers specify video transformer architectures and clip sizes.

In the case of videos, global self-attention becomes especially costly, since the number of patches to consider scales linearly with the number of frames T in the clip. Thus, the cost becomes quadratic in the space-time resolution, i.e., $\mathcal{O}(S^2T^2)$. Exhaustive comparison of all space-time pairs of patches in a video is not only prohibitively costly but also highly redundant, since adjacent frames are likely to contain strongly correlated information. One way to reduce the computational cost is to limit the temporal attention to compare only pairs of patches in different frames but at the *same* spatial location [2, 5]. The complexity becomes $\mathcal{O}(S^2T)$. However, this form of attention forces the model to perform temporal reasoning by looking through 1-dimensional "time tubes" of individual patches. Unfortunately, due to camera and object motion, the scene region projected onto a particular patch in frame t may be completely unrelated to the scene information captured in the same patch at a different time t'. For example, the object appearing in the patch at time t may have moved outside that patch in frame t', thus rendering the recovery of motion information impossible and making the comparison between the two patches

less relevant. Some works have proposed pooling tokens from local windows [10], or merging patches and limiting the self-attention operation to local space-time neighborhoods [29]. While effective in reducing the computational cost of video transformers, these methods employ fixed, hand-designed attention strategies that compare patches at predetermined locations, ignoring the motion in the video.

In this work we propose a novel space-time attention mechanism for video transformers, which we name *deformable space-time attention*. While previous works rely on fixed schemes of attention which neglect the dynamic nature of the video, our method leverages motion cues to determine which patches to compare, thus implementing a form of input-conditioned attention. Given a query patch at a certain space-time location, our method samples N patches to compare to that query patch in each of the other frames. The locations of these N samples are predicted according to motion cues relating the query patch to the frame to attend. In other words, our strategy decides dynamically "where to look" in each frame on the basis of the appearance of the query patch and its estimated motion. Importantly, the motion cues are obtained at zero cost directly from the information stored in the compressed format of the video, specifically in terms of *motion displacements* encoding the motion between adjacent frames. As a result, the computational cost of our deformable attention is only $\mathcal{O}(ST^2N)$. In our experiments we demonstrate that a small number of sample locations N is sufficient to achieve strong performance (e.g., most of our results are obtained with $N = 8$). This implies that, in practice, the complexity of our deformable attention is considerably lower than that of global attention, which is $\mathcal{O}(S^2T^2)$ where $S >> N$. In addition to the efficiency benefits, our experiments show that our deformable attention achieves higher accuracy than global attention, attaining state-of-the-art results on four action classification benchmarks.

2. Related Work

Image and Video Transformers: The transformer architecture was originally introduced by Vaswani et al. [39] as a model to capture long-range interactions between words in a sentence and was demonstrated to yield powerful representations for a variety of NLP tasks when trained with reconstruction objectives [7]. To extend transformers to the vision domain, Dosovitskiy et al. [9] proposed the Vision Transformer (ViT), which applies a linear projection to tokenize non-overlapping patches of the image and then feeds the resulting tokens to a sequence of multi-headed self-attention blocks. Touvron et al. [36] leveraged data augmentations and a student-teacher training scheme to increase the data efficiency of ViT. More recently, we have seen several attempts at extending ViT to the video domain [1, 2, 5, 10, 31]. TimeSformer [5], ViViT [2] and VideoSwin [29] treat time as an extra dimension and propose several temporal attention schemes to compare patches in different frames. MViT [10] reduces significantly the computational cost of video transformers by means of a local pooling operation that reduces progressively the number of tokens while increasing the channel dimension. The attention scheme that is most closely related to our own is the "trajectory attention" of MotionFormer, which is introduced in concurrent work [31]. This approach shares the same motivation as our strategy. However, it differs in two fundamental aspects. First, while our method leverages motion information available "for free" in the compressed video, trajectory attention requires the costly computation of a correlation volume expressing the probabilistic trajectories of each query patch over the clip. This causes trajectory attention to have quadratic cost $\mathcal{O}(S^2T^2)$ and to be, in practice, twice as costly as global attention in terms of GFLOPs. Second, trajectory attention uses a hand-designed attention strategy giving more importance to tokens along the motion trajectory of the query patch. Conversely, our model uses the motion displacement (plus appearance information) as cues to predict "where to look." However, the actual attention model is *learned* by optimizing the end-objective of video classification instead of being hand-defined.

Efficient Attention Schemes: Several solutions have been proposed to reduce the quadratic cost of global self-attention. One direction is to introduce manually defined local attention windows, and to restrict the attention operation to query-key pairs in the same local window. Longformer [4] and BigBird [45] implement this idea in the NLP domain, while Swin [28, 29] and CSwin Transformer [8] apply it to vision problems. Local windows, however, limit the span of the receptive field. To contrast this limitation, Longformer [4] and BigBird [45] allow a fraction of the tokens to use global attention, while Swin [28, 29] and CSwin Transformer [8] propose spatial merging and window shifting to increase the receptive field. In the NLP domain, Reformer [21] and Routing Transformer [32] use locality-sensitivity hashing and k-means to find the most relevant keys from all candidates. Deformable DETR [48] follows a similar idea for the problem of object detection in still images. It uses a learned attention function to select the most salient patches to compare to the given query patch. The locations of the patches to attend are predicted from multi-scaled feature maps obtained with an independently-trained CNN. Our Deformable Video Transformer borrows the sampling strategy of Deformable DETR and extends it to the video domain by leveraging motion cues stored in compressed video to identify the most salient patches in frames different from the query patch. We also evaluate an optional multi-scale deformable attention to compare patches within the same frame. Differently from Deformable DETR, our multi-scale feature maps are obtained

on the fly by means of a learnable convolutional layer attached to the attention block. This makes it possible to optimize the multiscale representation end-to-end together with the rest of the model with respect to the objective of video classification. Our experiments demonstrate that multi-scale attention provides synergistic information to that captured by deformable spatiotemporal attention, and that when these two attentions are combined together they lead to state-of-the-art results.

Temporal Modeling in Video: Analysis of the temporal information in the video is obviously critical for accurate action classification. Modern deep video architectures usually perform temporal analysis either by incorporating layers that can extract motion cues from raw RGB frames, such as 3D convolutions [17, 19, 37, 38, 43] and other learnable temporal operators [16, 24, 26, 34, 40], or by feeding pre-extracted optical flow to the network [14, 33, 41]. More related to our own approach, a few works have proposed to leverage motion cues already available in compressed video (such as motion displacements stored by modern codecs, e.g., MPEG-4 and H.264) for the purpose of action recognition [18, 22, 42, 46]. In this paper we also make use of these "free" motion cues in compressed video. However, instead of treating them as an additional input modality for action classification, we employ them to guide the spatiotemporal sampling of patches for self-attention.

3. Technical Approach

We introduce our approach by first reviewing the general architecture of video transformers, which adapt the still-image framework of the Vision Transformer (ViT) [9] to the video setting. We then describe our space-time deformable attention, which replaces the manually-defined attention mechanism of prior video transformers with an input-dependent, dynamic attention strategy that results in lower computational cost and higher accuracy.

3.1. Background: Video Transformers

Our method is designed to operate with any video transformer framework [2, 5, 10, 30, 31] and is not tied to a specific architecture. A video transformer takes as input a video clip $X \in \mathbb{R}^{H\times W\times 3\times T}$ consisting of T RGB frames sampled from the original video. The video is converted into a sequence of $S \cdot T$ tokens $\mathbf{x}_s^t \in \mathbb{R}^D$ for $s = 1, \ldots, S$ and $t = 1, \ldots, T$. The tokens $\mathbf{x}_s^t$ are obtained by decomposing each frame into S patches which are then projected to a D-dimensional space through a learnable linear transformation. This tokenization can be implemented by linearly mapping the RGB patches of each frame [5, 30] or by projecting space-time cubes of adjacent frames [2, 10, 31]. Separate learnable positional encodings $\mathbf{e}_s$ and $\mathbf{e}^t$ are then applied to the patch embeddings $\mathbf{x}_s^t$ for the spatial and the temporal dimension: $\mathbf{z}_s^t = \mathbf{x}_s^t + \mathbf{e}_s + \mathbf{e}^t$.

The tokens are processed by a sequence of L transformer layers applying multi-head attention (MHA) [39], layer norm (LN) [3], and multilayer perceptron (MLP) computation. Formally, layer ℓ transforms a token $\mathbf{z}_s^{t\,(\ell)}$ into token $\mathbf{z}_s^{t\,(\ell+1)}$ according to the following updates:

$$\hat{\mathbf{z}}_s^{t^{(\ell+1)}} = \text{MHA}\left(\text{LN}\left(\mathbf{z}_s^{t^{(\ell)}}\right)\right) + \mathbf{z}_s^{t^{(\ell)}} \tag{1}$$

$$\mathbf{z}_s^{t^{(\ell+1)}} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}_s^{t^{(\ell+1)}}\right)\right) + \hat{\mathbf{z}}_s^{t^{(\ell+1)}} \tag{2}$$

where $\ell = 1, \ldots, L$ and $\mathbf{z}_s^{t^{(0)}} = \mathbf{z}_s^t$.

While most video transformers maintain a constant space-time resolution $S \times T$ and fixed feature dimension D across all layers, some works have proposed patch-pooling [10] and patch-merging [29], which progressively reduce the space-time sequence length.

In order to lighten the notation, we describe the self-attention operation in the simplified setting of a single head of attention and drop the layer superscripts. First, query-key-value vectors for each patch at space location s and time instant t are computed via linear projections:

$$\mathbf{q}_s^t = W_q \text{LN}\left(\mathbf{z}_s^t\right) \tag{3}$$

$$\mathbf{k}_s^t = W_k \text{LN}\left(\mathbf{z}_s^t\right) \tag{4}$$

$$\mathbf{v}_s^t = W_v \text{LN}\left(\mathbf{z}_s^t\right) \tag{5}$$

where W_q W_k and W_v are matrices of learnable parameters. Next, the self-attention operation computes a weighted sum of the value vectors, where the combination coefficients are obtained by comparing the query to the keys.

Global space-time attention. In global attention (also known as dense or "joint space-time" [5] attention), each query patch is compared to all the other patches in the clip:

$$\hat{\mathbf{z}}_s^t = \mathbf{z}_s^t + \sum_{s'=1}^{S}\sum_{t'=1}^{T} \alpha_{s,s'}^{t,t'} \mathbf{v}_{s'}^{t'} \tag{6}$$

where the coefficients $\alpha_{s,s'}^{t,t'}$ are computed in terms of dot products between the query and the keys, and are normalized to sum up to 1:

$$\alpha_{s,s'}^{t,t'} = \frac{\exp < \mathbf{q}_s^t, \mathbf{k}_{s'}^{t'} >}{\sum_{s'',t''} \exp < \mathbf{q}_s^t, \mathbf{k}_{s''}^{t''} >}. \tag{7}$$

As can be seen from Eq. (7), the computational complexity of global space-time attention is $\mathcal{O}(S^2T^2)$.

Divided space-time attention. In order to reduce the computational cost and to lighten the memory footprint, prior works [2, 5] have proposed to factorize the attention operation along the temporal and spatial dimensions. **Space** attention compares only patches within the same frame t:

$$\hat{\mathbf{z}}_s^t = \mathbf{z}_s^t + \sum_{s'=1}^{S} \alpha_{s,s'}^{t} \mathbf{v}_{s'}^{t}. \tag{8}$$

Conversely, **time** attention compares the query patch only to patches at the same spatial location s in the other frames:

$$\hat{\mathbf{z}}_s^t = \mathbf{z}_s^t + \sum_{t'=1}^{T} \alpha_s^{t,t'} \mathbf{v}_s^{t'}. \tag{9}$$

The computational costs of these factorized operations are dramatically lower than that of dense attention: $\mathcal{O}(S^2T)$ for space-only attention, and $\mathcal{O}(ST^2)$ for time-only attention. In order to allow the model to capture dependencies along both dimensions, space attention and temporal attention are combined within each layer either by composition [5] or by concatenation of their independent outputs [2].

The fundamental limitation of factorized self-attention is that the temporal attention module compares patches in different frames but at the *same* spatial location. However, due to camera and object motion, the same patch will capture disparate 3D scene regions at different times in the sequence. Temporal attention encodes appearance correlations between these different scene regions neglecting to take into account the dynamics of the scene.

3.2. Deformable Video Transformer

In this subsection we introduce our Deformable Video Transformer, which uses motion cues to identify a sparse set of space-time location to attend for each query. Our motion cues are obtained at zero computational cost by leveraging motion displacements stored in the compressed format of the video and our learned attention module is directly optimized with respect to the end-objective of classification.

Deformable Space-Time Attention (D-ST-A). Our proposed scheme resembles time-only attention but with two fundamental differences. The first is that, for each query $\mathbf{q}_s^t$, the method attends N locations $\{s(n)\}_{n=1}^N$ within each frame t', instead of a single location s. Thus, the computational cost becomes $\mathcal{O}(ST^2N)$. This is dramatically lower than the $\mathcal{O}(S^2T^2)$ complexity of dense space-time attention, since N is more than two orders of magnitude smaller than the number of spatial patches S (we use $N = 8$ while $S = 3,136$). At the same time, attending multiple locations per frame allows the method to capture a bigger spatial context compared to vanilla time-only attention.

The second key difference is that these N locations are input-conditioned, i.e., their positions are computed as a function of $\mathbf{q}_s^t$ *and* of the motion relating frame t to t'. The intuition is that motion cues between the two frames provide salient information about where to look (i.e., which locations to attend) in frame t'. See Figure 2 for a visualization of attended patches in frame $t-\delta$, t, and $t+\delta$, given a query in frame t. We refer the reader to the supplementary material for additional visualizations.

Formally, the update equation for the token at location s in frame t becomes:

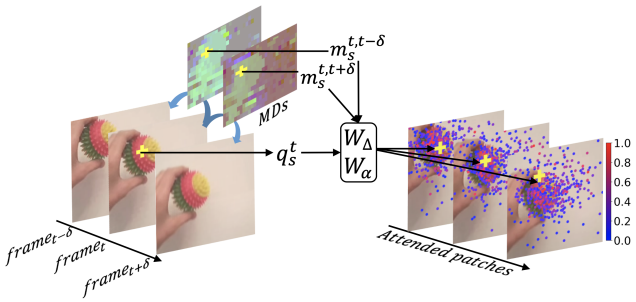

Figure 2. A visualization of D-ST-A. The yellow cross denotes a query patch in frame t. Each attention layer samples $N = 8$ patches with associated attention weights from each frame based on motion displacements (MDs). Filled circles denote sampled patch centers and their color (from blue to red) indicates the value of the attention weights (from 0 to 1). Additional examples of attended patches are given in the supplementary material.

$$\hat{\mathbf{z}}_s^t = \mathbf{z}_s^t + \sum_{t'=1}^{T} \sum_{n=1}^{N} \alpha_{s(n)}^{t,t'} \mathbf{v}_{s(n)}^{t'}. \tag{10}$$

The value vector $\mathbf{v}_{s(n)}^{t'}$ is computed via bilinear interpolation at 2D location $\mathbf{p}(s(n))$, which is obtained by adding an offset $\Delta_s^{t,t'}(n)$ to the 2D coordinates $\mathbf{p}(s)$ of patch s:

$$\mathbf{p}(s(n)) = \mathbf{p}(s) + \Delta_s^{t,t'}(n). \tag{11}$$

The offset $\Delta_s^{t,t'}(n)$ is computed by means of a linear projection of the *appearance* embedding $\mathbf{z}_s^t$ and the *motion* embedding $\mathbf{m}_s^{t,t'}$:

$$\Delta_s^{t,t'} = W_\Delta(\mathbf{q}_s^t + \mathbf{m}_s^{t,t'}) \tag{12}$$

where $W_\Delta \in \mathbb{R}^{2N \times D}$ is a matrix of learnable parameters.

To compute the motion embedding $\mathbf{m}_s^{t,t'}$ we utilize motion displacements and RGB residuals stored in the compressed video, which can both be easily accessed without any additional computation. Specifically, codecs such as MPEG-4 and H.264 represent compressed video in the form of sparsely stored I-frames, each followed by a sequence of 11 P-frames. An I-frame represents a frame in explicit RGB format at regular resolution, while the following P-frames encode implicitly the 11 subsequent RGB frames in terms of motion displacements (MD) and RGB residuals (RGB-R) which capture motion between adjacent frames and remaining RGB differences at low resolution, respectively. The RGB frames after an I-frame can be approximately reconstructed by rewarping the information stored in the I-frame according to the MD motion fields and by adding the remaining RGB-Rs. In this work, we accumulate the motion displacements and the RGB residuals between time step t and t' to represent the cumulative motion

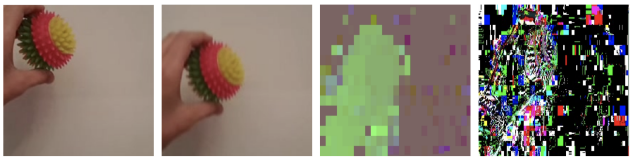

Figure 3. Example of motion displacements (MDs) and RGB residuals (RGB-Rs) between $frame_t$ and $frame_{t'}$.

between these two frames from the information stored for the adjacent frames in-between. We then apply a patch decomposition and a learnable tokenization separately to the MDs and the RGB-Rs in order to obtain motion embeddings $\mathbf{m}_s^{MD^{t,t'}}, \mathbf{m}_s^{RGBR^{t,t'}} \in \mathbb{R}^D$. In the experiments we compare quantitatively these two different motion embeddings. Examples of motion displacements and RGB residuals are given in Figure 3.

Note that in our system the attention coefficients are linearly projected from the appearance and motion embeddings instead of being computed by pairwise comparisons of keys and values. Specifically, they are obtained by first computing the vector of coefficients $\hat{\boldsymbol{\alpha}}_s^{t,t'} \in \mathbb{R}^N$:

$$\hat{\boldsymbol{\alpha}}_s^{t,t'} = W_\alpha(\mathbf{q}_s^t + \mathbf{m}_s^{t,t'}) \tag{13}$$

with $W_\alpha \in \mathbb{R}^{N \times D}$ and, finally, by normalizing the N coefficients per sample location:

$$\alpha_{s(n)}^{t,t'} = \frac{\exp \hat{\alpha}_{s(n)}^{t,t'}}{\sum_{t'=1}^{T} \sum_{n'=1}^{N} \exp \hat{\alpha}_{s(n')}^{t,t'}}. \tag{14}$$

While the self-attention coefficients could also be obtained by means of a traditional dot-product between queries and keys, the linear projection has been shown to give equivalent results at lower computational cost [47, 48].

We found empirically that motion embeddings $m_s^{t,t'}$ become noisy proxies of motion paths when the temporal distance $(t - t')$ between the two frames is large, due to error accumulation as well as visual nuisances such as camera movements, scenes changes and occlusions. To address this problem, we subdivide the input clip of T frames into B non-overlapping sub-clips, each containing $T' = T/B$ frames. We then restrict the accumulation of value vectors in Eq. 10 to include only terms from frames t' that belong to the same sub-clip as the query. Our experiments include an ablation showing the effects of different values of B on the recognition accuracy.

Deformable Multi-Scale Attention (D-MS-A). Inspired by the strong benefits shown by the use of multi-scale feature maps in transformer architectures [10, 48], we integrate an optional deformable multi-scale attention (D-MS-A) in our design. While our deformable *space-time* attention is used to capture dependencies between different frames by means of motion embeddings, our deformable *multi-scale* attention is a form of "space-only" attention encoding correlations within the same frame as the query. For each frame t, this involves computing F feature maps $\mathbf{z}_s^{(f),t}$ at different spatial resolutions $S^{(f)}$ for $f = 1, \ldots, F$, and then sampling N' patches to attend at each scale. The multi-scale feature maps are obtained by attaching a learnable 3D convolutional layer to each block. Then, the token update equation is given by:

$$\hat{\mathbf{z}}_s^t = \mathbf{z}_s^t + \sum_{f=1}^{F} \sum_{n=1}^{N'} \alpha_{s^{(f)}(n)}^t \mathbf{v}_{s^{(f)}(n)}^{(f),t} \tag{15}$$

where, as in the case of deformable space-time attention, the 2D coordinates $\mathbf{p}(s^{(f)}(n))$ of the n-th patch to attend at scale f are obtained by adding an estimated offset to the 2D location of the query at that scale, $\mathbf{p}(s^{(f)})$. Thus, $\mathbf{p}(s^{(f)}(n)) = \mathbf{p}(s^{(f)}) + \Delta_s^{(f),t}(n)$. However, this time the offset is predicted from the features at scale f for the *same* frame as the query:

$$\Delta_s^{(f),t} = W_\Delta^{(f)} \mathbf{q}_s^{(f),t}. \tag{16}$$

Attention Fusion. In our experiments we present results using only deformable space-time attention (D-ST-A), only deformable multi-scale attention (D-MS-A), as well as a combination of the two (D-ST+MS-A). In the last case, our method feeds the two tokens, $\mathbf{z}^{ST^t}{}_s$ and $\mathbf{z}^{MS^t}{}_s$, computed independently by these two attention strategies to an *attention fusion* layer $u()$ which produces the token $\mathbf{z}_s^t$ to pass to the next block: $\mathbf{z}_s^t = u(\mathbf{z}^{ST^t}{}_s, \mathbf{z}^{MS^t}{}_s)$. We present results for two forms of attention fusion, one based on a simple linear projection and the other based on the MLP-Mixer module [35]. The architecture details of these two modules are given in the supplementary material.

Multiple Heads of Attention. As customarily done in transformers, we use multiple heads of attention in each layer, allotting an equal number of dimensions to each head. The outputs of the individual heads are concatenated and projected through an MLP with a residual connection.

4. Experiments

4.1. Implementation Details

To demonstrate the generality of our Deformable Video Transformer (DVT), we apply it to two open-source video transformer architectures—MViT [10] and TimeSformer [5]—where we substitute their traditional fixed attention strategies with our input-dependent deformable attention. We refer to these two architectures as MViT-B and ViT-B, respectively.

Unless otherwise noted, we use input clips of size $16 \times 224 \times 224$, sampled with temporal stride of 4. The clip-level

classification is obtained by feeding the average-pooled features from the last layer to a fully connected layer. Video-level classification is performed by averaging the clip-level predictions computed from uniformly-spaced clips sampled from the video. The optimization recipe and the data augmentation strategy follow the implementation in [10]. For our deformable attention, we set $N = 8$ and $B = 4$ in both D-ST-A and D-MS-A by default, but we further ablate these hyperparameters in the experiments. In D-MS-A, to generate multi-scale feature maps we adopt a 3D convolutional layer with kernel size $3 \times 3 \times 3$, initial spatial stride of 8 (halved at each stage) and temporal stride of 1.

4.2. Datasets

We evaluate our DVT on four standard video classification benchmarks: Kinetics-400 [20] (K400), Something-Something-V2 [15] (SSv2), EPIC-KITCHENS-100 [6] (EK100), and Diving-48 [25] (D48).

4.3. Ablation Studies

We begin by studying how the accuracy varies as we modify the hyperparameters and the design choices in our DVT. Due to the high computational cost involved by these numerous ablations, here we limit these evaluations to the MViT architecture on the K400 and SSv2 benchmarks. The model on K400 is trained from random initialization (scratch). Since SSv2 is a much smaller dataset, we report results on it by finetuning the model pretrained on K400.

Choice of motion cues. In Table 1 we study the effectiveness of two different motion cues obtained from compressed video: RGB residuals (RGB-Rs) and motion displacements (MDs). We also consider a couple of simple cues directly computable from RGB values: "Averaged Pooled RGB" (Avg-P-RGB) averages the RGB values of the patches in query position s between all frames between time t and time t'; "RGB Differences" (RGB-D) uses the pixel-wise differences between the patches in query position s between frame t and frame t' . We include also results obtained with TV-L1 Optical Flow [44] (OF) since optical flow represents the ideal motion cue to use, neglecting cost considerations. Table 1 shows that Avg-P-RGB is the worst cue. RGB-D and RGB-R perform similarly and only slightly better than Avg-P-RGB. Conversely, MDs yield much higher accuracy than RGB-Rs, nearly on par with OF. This makes sense, since Avg-P-RGB, RGB-R and RGB-D capture color information which is only loosely related to the motion (e.g., as shown in Figure 3, RGB-R values tend to be higher in regions affected by motion, due to the need to fill-in the color of these regions to accurately reconstruct the RGB frames). Conversely, MDs encode true motion information between adjacent frames. Given the high computational cost of extracting OF, MD is an effective surrogate, achieving similar performance at no cost. Based on these results, we use MDs for all our subsequent experiments.

Motion Cue	K400	SSv2
Averaged Pooled RGB (Avg-P-RGB)	72.3%	62.9%
RGB Residuals (RGB-R)	73.2%	63.3%
RGB Differences (RGB-D)	73.6%	63.5%
Motion Displacements (MD)	77.9%	65.6%
Optical Flow (OF)	78.2%	66.0%

Table 1. Accuracy of DVT with D-ST-A attention on K400 and SSv2 using different motion cues.

Attention	K400	SSv2	GFLOPs	Parameters
MHPA [10]	78.4%	64.7%	70.5	36.6M
D-MS-A	76.8%	65.1%	61.3	32.4M
D-ST-A	77.9%	65.6%	57.4	31.9M
D-ST+MS-A (L)	78.1%	66.8%	81.9	45.1M
D-ST+MS-A (M)	**79.0%**	**67.5%**	128.3	73.9M

Table 2. Accuracy of DVT on K400 and SSv2 for our two proposed attention schemes (D-ST-A and D-MS-A) as well as their combination (D-ST+MS-A) using either a linear projection (L) or MLP-Mixer [35] (M) as fusion block. The architecture is MViT-B.

Attention and fusion blocks. In Table 2 we compare the performance of our two proposed attention schemes (D-ST-A and D-MS-A) as well as their combination (D-ST+MS-A) using either a linear projection (L) or MLP-Mixer [35] (M) as fusion block. We also include the results obtained with the original Multi Head Pooling Attention (MHPA) proposed in MViT [10]. Compared to MHPA, DVT with simple D-ST-A has lower cost (57.4 vs 71 GFLOPs) and yields better accuracy on SSV2 (65.6% vs. 64.7%). On K400 is nearly on-par in terms of accuracy (77.9% vs 78.4%) while providing an efficiency gain. Merging D-MS-A with D-ST-A using our fusion module elevates substantially the accuracy. The best results are obtained with MLP-Mixer [35] as fusion block on both K400 and SSv2, although at a higher cost. MLP-Mixer, specifically Channel-Mixer, projects the features into a higher dimensional space with a non-linear activation function, which allows better channel-wise communication compared to the linear projection. Based on these results, we use D-ST+MS-A with MLP-Mixer for the rest of the experiments.

Number of patches in the deformable attention. In Figure 4a we present the accuracy of our DVT model for a varying number of attended patches, N, in each frame, using D-ST+MS-A as attention. We see that while accuracy increases when moving from $N = 4$ to $N = 8$, the accuracy tends to level off after $N = 8$. This confirms that the frames in the video contain highly redundant information and that it is sufficient to attend a small number of patches to achieve strong accuracy. Since this hyperparameter affects the computational cost of our model, we set $N = 8$ for the subsequent studies as this represents a good trade-off in terms of accuracy versus efficiency.

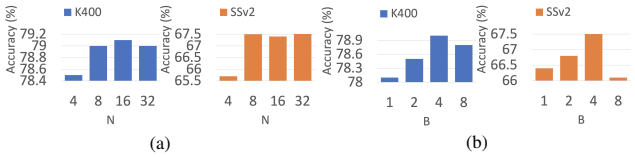

Figure 4. Accuracy of DVT on K400 and SSv2 when varying (a) the number of attended patches in each frame, N, or (b) the number of subclips for interframe comparison, B. The attention strategy is D-ST+MS-A (M) and the architecture is MViT-B.

Number of sub-clips in D-ST-A. In Figure 4b we report accuracy achieved when varying the number of attention sub-clips B in each clip. Each sub-clip contains $T' = T/B$ frames where T is the number of frames in the clip (here we use $T = 16$). D-ST-A compares only frames belonging to the same sub-clip. We see that the best accuracy is achieved for $B = 4$, which entails comparing $T' = 4$ frames within each sub-clip. Using smaller values for B yields lower accuracy as motion displacements become unreliable on longer sub-clips. Conversely, using $B = 8$ limits the ability of D-ST-A to leverage temporal information, which is particularly detrimental on SSv2.

Per-Class Comparison. In Table 3, we show the five K400 classes that receive the largest gain/degradation in accuracy from the use of D-ST-A compared to the same model (MViT) using MHPA. It can be seen that D-ST-A delivers much bigger gains for the top-5 classes, compared to the small drops in accuracy on the five classes where D-ST-A is most detrimental.

Improved Classes	Change	Degraded Classes	Change
Eating Doughnuts	+40.8%	Busking	-8.0%
Using Controller	+40.7%	Dining	-6.1%
Drinking Shots	+37.5%	Diving Cliff	-6.1%
Passing Football	+32.7%	Pumping Gas	-4.2%
Peeling Potatoes	+30.0%	Motorcycling	-4.1%

Table 3. K400 classes receiving the largest gain/degradation in accuracy when inserting D-ST-A in the MViT architecture.

4.4. Comparison to the state-of-the-art

In Tables 4, 5, 6, 7 we compare our method against the state-of-the-art on K400, SSv2, EK100, and D48. We note that the accuracy of an action recognition model depends strongly on (i) the architecture, (ii) the pretraining dataset, (iii) the clip size. In order to make the comparison with prior works as fair as possible, we evaluated our attention model under the many choices for (i, ii, iii) from the literature. To facilitate the interpretation of results, we split each Table into blocks, so that entries within the same block are results obtained with comparable (i, ii, iii) setups and differing with respect to the attention scheme (our contribution). The rows in bold are the results of our method.

Based on the results in Tables 4, 5, 6, 7, we can draw several observations. First, for the case of the MViT-B architecture, we can compare results obtained with our deformable attention against those achieved by the original Multi Head Pooling Attention (MHPA) proposed in MViT [10]. Compared to MHPA, our D-ST+MS-A provides a Top-1 accuracy gain of $+0.6\%$ on K400 (79.0% vs 78.4% in the learning from scratch setting) and of $+2.8\%$ on SSv2 (67.5% vs 64.7% when both using K400 pretraining). SSv2 is a benchmark designed to require effective temporal modeling. The larger gain on SSv2 compared to K400 suggests that our model beneficially leverages the motion cues to capture relevant temporal dependencies. In the case of the ViT-B architecture, we can compare our deformable attention against Divided-ST attention of TimeSformer [5] and Trajectory attention of MotionFormer [31], since both these works reported results using this architecture. Compared to Trajectory attention, our D-ST+MS-A achieves an accuracy gain of $0.8 - 0.9\%$ on K400 (80.5% vs 79.7% for 16×224^2 clips and 82.0% vs 81.1% for 16×336^2 clips). Furthermore, our DVT is 7 times more efficient than MotionFormer in terms of GFLOPs. Our D-ST+MS-A improves over Trajectory attention by $+0.5 - 0.8\%$ on SSv2 (67.0% vs 66.5% for 16×224^2 and 67.9% vs 67.1% for 16×336^2 clips) and by $+0.7\%$ on Action prediction (43.8% vs 43.1%) $+1.2\%$ on Verb prediction in EK100 ($68.2.8\%$ vs 67.0%) when both models use the same clip size. Compared to Divided-ST, our D-ST+MS-A produces an accuracy gain of $+3.8\%$ on K400 (84.5% vs 79.7%) when both use a clip size of 16×448^2. On D48, which is another benchmark designed to assess temporal modeling abilities, D-ST+MS-A yields an accuracy gain of $+6.6\%$ over Divided-ST (86.6% vs 78.0%) given the same input resolution. Finally, we also include results of DVT with the MViT-B architecture using longer clips (32×224^2) which allow our method to achieve further improvements over the state-of-the-art on all four benchmarks.

5. Conclusion

We have introduced a novel self-attention strategy for video classification. Unlike prior schemes that compare patches at predetermined locations, our method leverages motion cues to identify the most salient patches to compare to each query patch. The motion cues are obtained at zero cost from compressed video. This allows our method to achieve higher accuracy and lower computational cost compared to fixed schemes of attention. Currently, our approach limits the temporal span of attention to prevent motion errors to degrade the results. In the future, we are interested in exploring alternative motion cues that can be computed efficiently [16, 24, 26, 34] and that may provide more reliable estimates of motion over longer temporal extents.

Method	Attention	ViT Architecture	Pretraining	Clip Size	Top-1	Top-5	GFLOPs x views
SlowFast [13]	-	-	-	8×224^2	79.8%	93.9%	$234 \times 3 \times 10$
X3D [12]	-	-	-	16×312^2	79.1%	93.9%	$48 \times 3 \times 10$
Tformer [5]	Divided-ST	ViT-B	IN-21K	8×224^2	78.0%	93.7%	$197 \times 3 \times 1$
Mformer [31]	Trajectory	ViT-B	IN-21K	16×224^2	79.7%	94.2%	$397 \times 3 \times 10$
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-21K**	$\mathbf{16 \times 224^2}$	**80.5%**	**94.7%**	$\mathbf{325 \times 1 \times 5}$
Mformer [31]	Trajectory	ViT-B	IN-21K	16×336^2	81.1%	95.2%	$959 \times 3 \times 10$
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-21K**	$\mathbf{16 \times 336^2}$	**82.0%**	**95.3%**	$\mathbf{731 \times 1 \times 5}$
Tformer [5]	Divided-ST	ViT-B	IN-21K	16×448^2	79.7%	94.4%	$1703 \times 3 \times 1$
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-21K**	$\mathbf{16 \times 448^2}$	**84.5%**	**96.7%**	$\mathbf{1300 \times 1 \times 5}$
Mformer [31]	Trajectory	ViT-B	IN-21K	32×224^2	80.2%	94.8%	$1185 \times 3 \times 10$
Tformer [5]	Divided-ST	ViT-B	IN-21K	96×224^2	80.7%	94.7%	$2380 \times 3 \times 1$
ViViT [2]	Joint ST	ViT-L	IN-21K	16×320^2	81.3%	94.7%	$3992 \times 3 \times 4$
MViT [10]	MHPA	MViT-B	-	16×224^2	78.4%	93.5%	$71 \times 1 \times 5$
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**-**	$\mathbf{16 \times 224^2}$	**79.0%**	**93.8%**	$\mathbf{128 \times 1 \times 5}$
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-1K**	$\mathbf{16 \times 224^2}$	**80.8%**	**95.0%**	$\mathbf{128 \times 1 \times 5}$
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-21K**	$\mathbf{16 \times 224^2}$	**81.5%**	**95.2%**	$\mathbf{128 \times 1 \times 5}$
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-21K**	$\mathbf{32 \times 224^2}$	**84.7%**	**96.2%**	$\mathbf{259 \times 1 \times 5}$

Table 4. Video-level classification accuracy on Kinetics-400.

Method	Attention	ViT Architect.	Pretraining	Clip Size	Top-1	Top-5
MSNet [23]	-	-	IN-1K	16×224^2	64.7%	89.4%
bLVNet [11]	-	-	IN-1K	32×224^2	65.2%	90.3%
Tformer [5]	Divided-ST	ViT-B	IN-1K	8×224^2	59.5%	74.9%
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-1K**	$\mathbf{8 \times 224^2}$	**64.8%**	**89.5%**
Tformer [5]	Divided-ST	ViT-B	IN-1K	16×448^2	62.2%	78.0%
Tformer [5]	Divided-ST	ViT-B	IN-1K	96×224^2	62.4%	81.0%
Mformer [31]	Trajectory	ViT-B	IN-21K+K400	16×224^2	66.5%	90.1%
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-21K+K400**	$\mathbf{16 \times 224^2}$	**67.0%**	**90.5%**
Mformer [31]	Trajectory	ViT-B	IN-21K+K400	16×336^2	67.1%	90.6%
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-21K+K400**	$\mathbf{16 \times 336^2}$	**67.9%**	**90.8%**
Mformer [31]	Trajectory	ViT-B	IN-21K+K400	32×224^2	68.1%	91.2%
ViViT [2]	Joint-ST	ViT-L	IN-21K	16×320^2	65.4%	89.8%
MViT [10]	MHPA	MViT-B	K400	16×224^2	64.7%	89.2%
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**K400**	$\mathbf{16 \times 224^2}$	**67.5%**	**90.8%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-1K**	$\mathbf{16 \times 224^2}$	**67.4%**	**90.6%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-21K**	$\mathbf{16 \times 224^2}$	**67.8%**	**90.6%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-1K+K400**	$\mathbf{16 \times 224^2}$	**67.8%**	**90.6%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-21K+K400**	$\mathbf{16 \times 224^2}$	**68.0%**	**91.0%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-21K+K400**	$\mathbf{32 \times 224^2}$	**68.5%**	**91.0%**

Table 5. Video-level classification accuracy on Something-Something-V2.

Method	Attention	ViT Architecture	Pretraining	Clip Size	A	V	N
TSM [27]	-	-	IN-1K	16×224^2	38.3%	67.9%	49.0%
Mformer [31]	Trajectory	ViT-B	IN-21K+K400	16×224^2	43.1%	66.7%	56.5%
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-21K+K400**	$\mathbf{16 \times 224^2}$	**43.8%**	**67.5%**	**56.5%**
Mformer [31]	Trajectory	ViT-B	IN-21K+K400	16×336^2	44.5%	67.0%	58.5%
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-21K+K400**	$\mathbf{16 \times 336^2}$	**45.0%**	**68.2%**	**58.7%**
Mformer [31]	Trajectory	ViT-B	IN-21K+K400	32×224^2	44.1%	67.1%	57.6%
ViViT [2]	Joint ST	ViT-L	IN-21K+K400	16×224^2	44.0%	66.4%	56.5%
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-1K**	$\mathbf{16 \times 224^2}$	**42.3%**	**68.4%**	**54.4%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-1K+K400**	$\mathbf{16 \times 224^2}$	**43.3%**	**68.9%**	**55.6%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-21K+K400**	$\mathbf{16 \times 224^2}$	**44.2%**	**69.7%**	**56.6%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-21K+K400**	$\mathbf{32 \times 224^2}$	**45.8%**	**69.9%**	**58.7%**

Table 6. Accuracy of Action (A), Verb (V) and Noun (N) classification achieved by different models on EPIC-KITCHENS.

Method	Attention	ViT Architecture	Pretraining	Clip Size	Top-1
SlowFast [13]	-	-	IN-1K	16×224^2	77.6%
Tformer [5]	Divided-ST	ViT-B	IN-1K	8×224^2	74.9%
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-1K**	$\mathbf{16 \times 224^2}$	**83.5%**
Tformer [5]	Divided-ST	ViT-B	IN-1K	16×448^2	78.0%
DVT (ours)	**D-ST+MS-A**	**ViT-B**	**IN-1K**	$\mathbf{16 \times 448^2}$	**86.6%**
Tformer [5]	Divided-ST	ViT-B	IN-1K	96×224^2	81.0%
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-1K**	$\mathbf{16 \times 224^2}$	**86.0%**
DVT (ours)	**D-ST+MS-A**	**MViT-B**	**IN-1K**	$\mathbf{32 \times 224^2}$	**86.9%**

Table 7. Video-level classification accuracy on the Diving-48 dataset.

References

[1] Hassan Akbari, Linagzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. *arXiv preprint arXiv:2104.11178*, 2021. 2

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6836–6846, October 2021. 1, 2, 3, 4, 8

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 2

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 813–824. PMLR, 2021. 1, 2, 3, 4, 5, 7, 8

[6] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Rescaling egocentric vision. *arXiv preprint arXiv:2006.13256*, 2020. The Epic-Kitchens-100 dataset is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License. 6

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[8] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arXiv preprint arXiv:2107.00652*, 2021. 2

[9] 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. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 3

[10] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6824–6835, October 2021. 2, 3, 5, 6, 7, 8

[11] Quanfu Fan, Chun-Fu Chen, Hilde Kuehne, Marco Pistoia, and David Cox. More is less: Learning efficient video representations by big-little network and depthwise temporal aggregation. *arXiv preprint arXiv:1912.00869*, 2019. 8

[12] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 203–213, 2020. 8

[13] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6202–6211, 2019. 8

[14] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1933–1941, 2016. 3

[15] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5842–5850, 2017. The Something-Something dataset is licensed for non-commercial, research purposes at https://developer.qualcomm.com/downloads/data-license-agreement-research-use?referrer=node/68935. 6

[16] Omar Hommos, Silvia L Pintea, Pascal SM Mettes, and Jan C van Gemert. Using phase instead of optical flow for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 3, 7

[17] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012. 3

[18] Vadim Kantorov and Ivan Laptev. Efficient feature extraction, encoding and classification for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 3

[19] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 3

[20] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. The Kinetics-400 dataset is licensed under the Creative Commons Attribution-NonCommercial 4.0 International License. 6

[21] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020. 2

[22] Bruno Korbar, Du Tran, and Lorenzo Torresani. Scsampler: Sampling salient clips from video for efficient action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6232–6242, 2019. 3

[23] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. In *European Conference on Computer Vision*, pages 345–362. Springer, 2020. 8

[24] Myunggi Lee, Seungeui Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 3, 7

[25] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 513–528, 2018. 6

[26] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3, 7

[27] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7083–7093, 2019. 8

[28] 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. *arXiv preprint arXiv:2103.14030*, 2021. 2

[29] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 2, 3

[30] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. *arXiv preprint arXiv:2102.00719*, 2021. 3

[31] Mandela Patrick, Dylan Campbell, Yuki M Asano, Ishan Misra Florian Metze, Christoph Feichtenhofer, Andrea Vedaldi, Jo Henriques, et al. Keeping your eye on the ball: Trajectory attention in video transformers. *Advances in neural information processing systems*, 2012. 1, 2, 3, 7, 8

[32] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. 2

[33] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*, 2014. 3

[34] Shuyang Sun, Zhanghui Kuang, Lu Sheng, Wanli Ouyang, and Wei Zhang. Optical flow guided feature: A fast and robust motion representation for video action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1390–1399, 2018. 3, 7

[35] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021. 5, 6

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 2

[37] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 3

[38] 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*, pages 6450–6459, 2018. 3

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 1, 2, 3

[40] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 352–361, 2020. 3

[41] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016. 3

[42] Chao-Yuan Wu, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Compressed video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6026–6035, 2018. 3

[43] 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 (ECCV)*, pages 305–321, 2018. 3

[44] Christopher Zach, Thomas Pock, and Horst Bischof. A duality based approach for realtime tv-l 1 optical flow. In *Joint pattern recognition symposium*, pages 214–223. Springer, 2007. 6

[45] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020. 2

[46] Bowen Zhang, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang. Real-time action recognition with enhanced motion vector cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3

[47] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 5

[48] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2, 5