Intervolution: Relational Transform for Video Understanding

Anonymous Author(s)
Affiliation
Address
email

Abstract

Convolution has been arguably the most important feature transform for modern neural networks, leading to the advance of deep learning. Recent emergence of attention-based networks, which replace the convolution layers with self-attention blocks, has revealed the limitation of stationary convolution kernels and opened the door to the era of dynamic feature transforms. The existing dynamic transforms, including self-attention, however, are all limited for video understanding where content-to-content relations in space and time, *i.e.*, motion information, are crucial for effective representation. In this work, we introduce a relational feature transform, dubbed *intervolution*, that captures rich spatio-temporal relations in videos by dynamically generating relational kernels and aggregating relational contexts. Detailed ablation studies demonstrate the advantages of intervolution over self-attention and recent dynamic transforms on the task of video action classification. Our intervolution network substantially outperforms convolution and self-attention counterparts, achieving the state of the art on the standard motion-centric benchmarks, Something-Something-V1&V2, Diving48, and FineGym.

1 Introduction

Convolution [1, 2] is a feature transform that is ubiquitous in modern neural networks for visual recognition, and has driven the development of deep learning in the past decade. The stationarity of convolution kernels, however, may limit its expressivity and hinder adaptation to diverse compositional possibilities of visual concepts [3]. Dynamic convolution [4] and self-attention [5] that construct kernels or attentions according to input contents have emerged as an alternative to static convolution in this context, being followed by further studies for dynamic feature transforms [6, 7, 8, 9]. The effectiveness of self-attention has been demonstrated by the success of Transformer variants on different image understanding tasks such as image classification [10, 11, 12], object detection [13], and semantic segmentation [14]. Recently, it has been further extended for video understanding, replacing spatio-temporal convolution [15, 16, 17] with spatio-temporal self-attention [18, 19, 20].

Despite their recent progress in the video domain, the existing dynamic feature transforms still leave much room for improvement in terms of learning relational patterns in space and time, *i.e.*, motion information, which is known to be essential for video understanding [21, 22]. For example, spatio-temporal self-attention [23] often fails to learn motion representation without positional embeddings as demonstrated in [24], and even those with positional embeddings [18, 19, 20] turn out to be not effective on motion-centric video understanding benchmarks such as Something-Something [25]. While existing dynamic feature transforms all use content-to-content relations, *i.e.*, correlations, for kernel construction, they fail to leverage the rich relational patterns in space and time.

In this work, we introduce a relational dynamic feature transform, dubbed *intervolution*, to address the limitation of existing methods. The intervolution captures rich motion features in videos by

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

dynamically generating relational kernels and aggregating relational contexts. The relational kernels are constructed by using inter-feature *relations* within a local spatio-temporal extent of input, while the relational context encodes spatio-temporal self-correlations within the input extent. Combining these components with ordinary dynamic kernels and context features, our intervolution learns a rich video representation that effectively captures both of visual information and spatio-temporal dynamics.

The main contribution of this paper is three-fold:

- We re-interpret recent dynamic feature transforms in a unified way, and provide in-depth analysis on their capability of learning video representation.
- We introduce a relational dynamic feature transform, *i.e.*, intervolution, which captures both visual appearance and spatio-temporal dynamics effectively for video understanding.
- Our intervolution network substantially outperforms convolution and self-attention counterparts, achieving the state of the art on SS-V1&V2, Diving48, and FineGym.

2 Related Work

Convolution and its variants. Convolution [1, 2] has been used as a dominant neural primitive in modern neural architectures [26, 27, 28, 29, 30, 31]. Since convolution often suffers from its limited expressivity due to its static kernel, dynamic convolution operators have been studied for enhancing composability by dynamically adjusting the convolution kernel according to input features [4, 6, 7, 8, 9]. One example is involution [9], which generates a lightweight dynamic kernel using input content and substantially outperforms convolution with less computational cost on image classification. Our intervolution differs from these kernel generation methods in that it leverages relational patterns for learning a rich video representation.

Self-attention and its variants. Self-attention [5] was originally introduced for neural machine translation to capture long-range interactions, and now has been widely adopted in many different domains thanks to its versatility and expandability. Following local attention methods [32, 3] that employ self-attention as an alternative to convolution, ViT models have demonstrated impressive performance on a variety of image understanding tasks [10, 11, 13, 14]. On the other hand, some of previous work have attempted to design novel dynamic transforms through attention [33, 34]. Zhao *et al.* [33] expand the spatial attention to include the channel-wise attention [33]. Bello [34] proposes a lambda layer that focuses on interaction between visual contents and a relative position embedding without softmax, which outperforms self-attention counterparts on image classification. The proposed intervolution is an extension of these techniques, yet focuses on learning rich relational features for video understanding.

Convolution and self-attention for action recognition. Action recognition is the task of classifying a short video clip to one of pre-defined action classes. A key to success of this task is to capture temporal dynamics across multiple video frames, and spatio-temporal (3D) convolution [15] has been a *de facto* for modeling the temporal dynamics [17, 35, 36, 37, 38]. On the other hand, self-attention was used as an attachable module for 3D CNNs in the earliest method [23], yet nowadays becomes the major building block of video understanding networks such as video ViT [18, 19, 20]. However, both 3D CNNs and ViTs are not sufficient for modeling temporal dynamics due to the lack of inter-feature relations within the spatio-temporal context. Our intervolution overcomes their limitation by explicitly utilizing relational features that imply temporal dynamics.

Learning motions for video understanding. Early approaches in video understanding rely on external motion extractors like optical flow for motion feature learning [16, 39, 40], but recent models aim to learn motion representation internally. To this end, they estimate feature-level optical flow inside their networks [41, 42] or compute subtraction between consecutive frame-wise features [43, 44, 45, 46]. Meanwhile, correlation-based methods [21, 22, 47] use inter-pixel relations as motion likelihood maps and achieve state of the art on motion-centric action classification benchmarks. Inspired by the correlation-based methods, we apply dynamic relational kernels to the correlation tensors (*i.e.*, relational context), leading to effective motion feature learning.

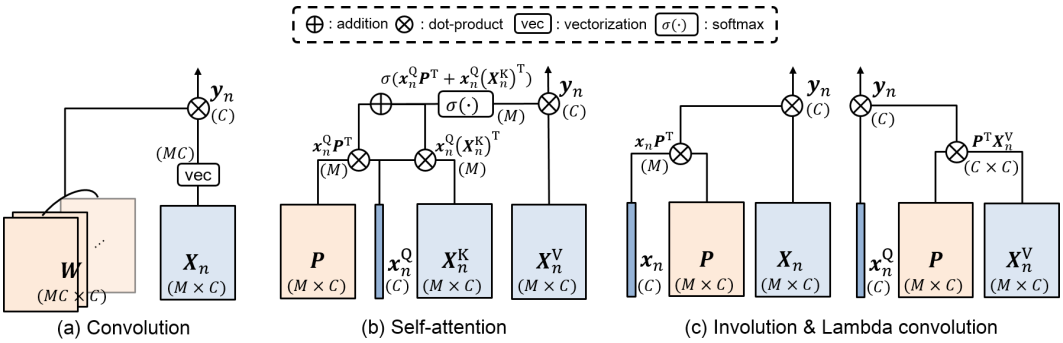

Figure 1: **Computational graph of (a) convolution, (b) self-attention, and (c) involution & lambda convolution.** See text for details.

3 Background

In this section, we revisit three instances of feature transform and analyze them in a simplified form. In our context, the role of feature transform is to update a feature map while preserving its order in space and/or time. To this end, it processes each position in space and/or time using features of its neighborhood. For example, given an input feature map of a video $\boldsymbol{X} \in \mathbb{R}^{T\times H\times W\times C}$, for each spatio-temporal position (t, h, w), a feature transform takes the individual feature $\boldsymbol{x}_{t,h,w}$ as a *target* and transforms it using its neighborhood features $\{\boldsymbol{x}_{t',h',w'}\}_{(t',h',w')\in\mathcal{N}_{(t,h,w)}}$ as its *context*. For the sake of notational convenience, let us denote the target by $\boldsymbol{x}_n \in \mathbb{R}^C$, where n represents a specific position in space and/or time, and its context by $\boldsymbol{X}_n \in \mathbb{R}^{M\times C}$, where M is the size of neighborhood, and the corresponding output by $\boldsymbol{y}_n \in \mathbb{R}^C$. Now, without loss of generality, we can view the *transform* as applying to each position n a function f with learnable weights $\boldsymbol{W}$ that maps target $\boldsymbol{x}_n$ to output $\boldsymbol{y}_n$ using context $\boldsymbol{X}_n$: $\boldsymbol{y}_n = f(\boldsymbol{x}_n, \boldsymbol{X}_n; \boldsymbol{W})$. Note that such a transform is typically implemented as a neural layer that processes all the targets in parallel.

In the following, we describe recent feature transform functions in computer vision as well as the conventional one, convolution. Figure 1 illustrates the computational graphs of the transforms.

Convolution. Convolution [1, 2] is a static and translation-equivariant feature transform that updates each target $\boldsymbol{x}_n$ by applying static position-specific kernels $\boldsymbol{W} \in \mathbb{R}^{MC\times C}$ on local context $\boldsymbol{X}_n$:

$$\boldsymbol{y}_n = \boldsymbol{W}^\mathsf{T}\mathrm{vec}(\boldsymbol{X}_n). \tag{1}$$

While extracting different visual patterns using multiple kernels, convolution remains static in the sense that the kernel is not affected by the target $\boldsymbol{x}_n$, *i.e.*, feature of interest (Figure 1a). This stationarity may not be effective in adapting to diverse compositional possibilities of visual concepts [3], and the channel-dependent weights can cause inter-channel redundancy in the kernel [48].

Self-attention. Self-attention [5] is a dynamic transform that generates an attention map of context $\boldsymbol{X}_n$ using target $\boldsymbol{x}_n$ and then aggregates the context using the attention map as a dynamic kernel (Figure 1b). The process starts by adapting inputs for query-key-value attention computation; using three learnable embedding matrices, $\boldsymbol{E}^\mathrm{Q}, \boldsymbol{E}^\mathrm{K}, \boldsymbol{E}^\mathrm{V} \in \mathbb{R}^{C\times C}$, it embeds the target $\boldsymbol{x}_n$ into the query $\boldsymbol{x}_n^\mathrm{Q}$ via $\boldsymbol{x}_n^\mathrm{Q} = \boldsymbol{x}_n\boldsymbol{E}^\mathrm{Q}$ and also projects the context $\boldsymbol{X}_n$ into the key $\boldsymbol{X}_n^\mathrm{K}$ and the value $\boldsymbol{X}_n^\mathrm{V}$ via $\boldsymbol{X}_n^\mathrm{K} = \boldsymbol{X}_n\boldsymbol{E}^\mathrm{K}$ and $\boldsymbol{X}_n^\mathrm{V} = \boldsymbol{X}_n\boldsymbol{E}^\mathrm{V}$, respectively. The basic attention map is computed via the softmaxed product between query $\boldsymbol{x}_n^\mathrm{Q}$ and key $\boldsymbol{X}_n^\mathrm{K}$, which can be viewed as *content-to-content interaction*. To reinforce this content-only attention with positional information of the context, a learnable matrix $\boldsymbol{P} \in \mathbb{R}^{M\times C}$, which encodes relative positions of individual features in the context, is optionally included as *content-to-position interaction*. The self-attention transform aggregates the value embeddings $\boldsymbol{X}_n^\mathrm{V}$ of context using the attention map as kernel weights shared for all channels:

$$\boldsymbol{y}_n = \sigma(\boldsymbol{x}_n^\mathrm{Q}(\boldsymbol{X}_n^\mathrm{K})^\mathsf{T} + \boldsymbol{x}_n^\mathrm{Q}\boldsymbol{P}^\mathsf{T})\boldsymbol{X}_n^\mathrm{V}, \tag{2}$$

where $\sigma(\cdot)$ denote the softmax function. This transform is translation-equivariant if position embedding $\boldsymbol{P}$ is local and relative with respect to the target, and becomes permutation-equivariant if content-to-position interaction is removed. Unlike convolution, it is *dynamic* in the sense that the kernel $\sigma(\boldsymbol{x}_n^\mathrm{Q}(\boldsymbol{X}_n^\mathrm{K})^\mathsf{T} + \boldsymbol{x}_n^\mathrm{Q}\boldsymbol{P}^\mathsf{T})$, which is used to aggregate the context, depends on the target content.

It allows more flexibility in adapting to input and also consumes fewer parameters by sharing the kernel across the channels [3].

Involution & lambda convolution. Involution [9] is a light-weight dynamic transform that leverages *content-to-position interaction* only. It dynamically generates a kernel $\boldsymbol{\kappa}_n^{\mathrm{V}} \in \mathbb{R}^M$ by projecting target $\boldsymbol{x}_n$ using a learnable matrix $\boldsymbol{P} \in \mathbb{R}^{M \times C}$ via $\boldsymbol{\kappa}_n^{\mathrm{V}} = \boldsymbol{x}_n \boldsymbol{P}^{\mathsf{T}}$, and then use the kernel to aggregate the context $\boldsymbol{X}_n$ (Figure 1c, left):

$$\boldsymbol{y}_n = \boldsymbol{\kappa}_n^{\mathrm{V}} \boldsymbol{X}_n = \boldsymbol{x}_n \boldsymbol{P}^{\mathsf{T}} \boldsymbol{X}_n, \tag{3}$$

where the matrix $\boldsymbol{P}$ plays the role of converting the target to a kernel in a position-sensitive manner. In a similar spirit, lambda convolution of the lambda layer [34] uses *content-to-position interaction* between the target and the context position. As in self-attention, using two learnable embedding matrices, $\boldsymbol{E}^{\mathrm{Q}}, \boldsymbol{E}^{\mathrm{V}} \in \mathbb{R}^{C \times C}$, it starts by projecting the target $\boldsymbol{x}_n$ and the context $\boldsymbol{X}_n$ into the query $\boldsymbol{x}_n^{\mathrm{Q}}$ and the value $\boldsymbol{X}_n^{\mathrm{Q}}$ via $\boldsymbol{x}_n^{\mathrm{Q}} = \boldsymbol{x}_n \boldsymbol{E}^{\mathrm{Q}}$ and $\boldsymbol{X}_n^{\mathrm{V}} = \boldsymbol{X}_n \boldsymbol{E}^{\mathrm{V}}$, respectively. The lambda convolution abstracts the context value $\boldsymbol{X}_n^{\mathrm{V}}$ to a contextual matrix $\boldsymbol{\lambda}_n^{\mathrm{p}}$ using a learnable matrix $\boldsymbol{P} \in \mathbb{R}^{M \times C}$ via $\boldsymbol{\lambda}_n^{\mathrm{p}} = \boldsymbol{P}^{\mathsf{T}} \boldsymbol{X}_n^{\mathrm{V}} \in \mathbb{R}^{C \times C}$, which in turn is used for updating the target query $\boldsymbol{x}_n^{\mathrm{Q}}$ (Figure 1c, right):

$$\boldsymbol{y}_n = \boldsymbol{x}_n^{\mathrm{Q}} \boldsymbol{\lambda}_n^{\mathrm{p}} = \boldsymbol{x}_n^{\mathrm{Q}} \boldsymbol{P}^{\mathsf{T}} \boldsymbol{X}_n^{\mathrm{V}}. \tag{4}$$

Despite the differences in the operational procedures and the concepts, the lambda convolution has effectively the same form with involution except for additional key and value embeddings. Note that unlike self-attention, involution and lambda convolution both have no softmax nonlinearity. In fact, the absence of softmax reduces the computational cost and also increases the expressive ability by allowing negative activations [9]. Both involution and lambda convolution are shown to outperform convolution and self-attention counterparts in image classification [9, 34], demonstrating the effectiveness of dynamic content-aware kernels.

Limitation of existing dynamic transforms. The aforementioned dynamic transforms are commonly based on leveraging content-to-content and/or content-to-position interactions in constructing kernels, where the target content (input feature) is used as the source of the dynamic transform. While the methods are effective for learning image representation indeed, they are all limited for learning video representation; as will be shown in the experimental section 5.3, we have found that existing dynamic transforms show only marginal or virtually no improvement over the static transform of convolution on the motion-centric action recognition benchmark. The main reason lies in the absence of content-to-content *interaction patterns* as a *relational content* in representation learning. While content-to-content interactions are considered in existing dynamic transforms, they are immediately broken into individuals without being used as a whole. For example, self-attention computes query-to-key correlations $\boldsymbol{x}_n^{\mathrm{Q}} (\boldsymbol{X}_n^{\mathrm{K}})^{\mathsf{T}}$ but uses the individual elements of correlation only for aggregating informative contents from the context. The content-to-position interaction $\boldsymbol{x}_n^{\mathrm{Q}} \boldsymbol{P}^{\mathsf{T}}$ does not help in capturing the interaction patterns either since it has no access to the correlation information. In videos, such interaction patterns in space and time contain informative spatio-temporal contents, *i.e.*, different levels of generic motion information, thus being crucial in video understanding.

4 Our approach

In this section we introduce a dynamic relational transform, dubbed *intervolution*, which is designed to learn rich spatio-temporal interaction patterns across input contents. Figure 2 illustrates the computational graph of the intervolution. On top of the basic kernel and context in a dynamic transform, it builds relational kernel and context and processes all the kernel-context combinations. In the following, we describe two major components of intervolution, *relational kernel* and *relational context*, and then explain the integrated form of intervolution and its implementation details.

As in self-attention, we start by adapting inputs for query-key-value interactions; using three learnable embedding matrices, $\boldsymbol{E}^{\mathrm{Q}}, \boldsymbol{E}^{\mathrm{K}}, \boldsymbol{E}^{\mathrm{V}} \in \mathbb{R}^{C \times C}$, target $\boldsymbol{x}_n$ and context $\boldsymbol{X}_n$ are embedded into query $\boldsymbol{x}_n^{\mathrm{Q}}$, key $\boldsymbol{X}_n^{\mathrm{K}}$ and value $\boldsymbol{X}_n^{\mathrm{V}}$ via $\boldsymbol{x}_n^{\mathrm{Q}} = \boldsymbol{x}_n \boldsymbol{E}^{\mathrm{Q}}$, $\boldsymbol{X}_n^{\mathrm{K}} = \boldsymbol{X}_n \boldsymbol{E}^{\mathrm{K}}$, and $\boldsymbol{X}_n^{\mathrm{V}} = \boldsymbol{X}_n \boldsymbol{E}^{\mathrm{V}}$, respectively. $\boldsymbol{x}_n^{\mathrm{Q}}$, $\boldsymbol{X}_n^{\mathrm{K}}$, and $\boldsymbol{X}_n^{\mathrm{V}}$ are then L2-normalized, and we omit the normalization term for the simplicity.

4.1 Relational kernel and relational context

Relational kernel. The relational kernel is designed to predict the relevance of context based on the relational pattern of content-to-content (*i.e.*, target-to-context) interactions. To perform the

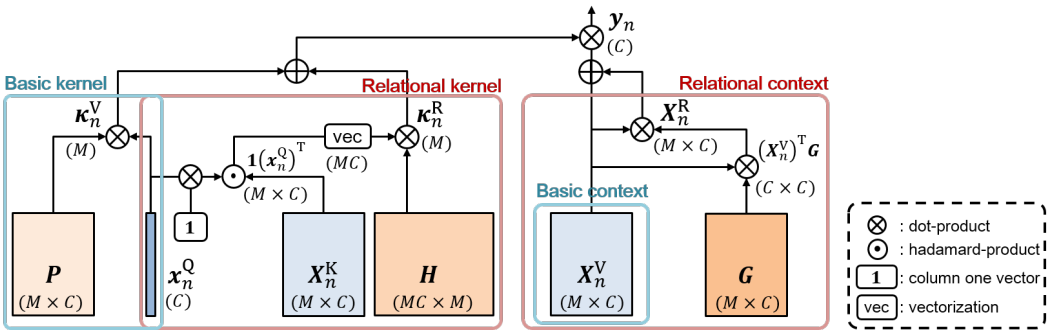

Figure 2: **Computational graph of Intervolution.** Intervolution consists of two types of kernels (basic and relational kernel) and two contexts (basic and relational context). See text for details.

interactions, we broadcast the query for the size of context via $\mathbf{1}(\boldsymbol{x}_n^{\mathrm{Q}})^{\mathsf{T}} \in \mathbb{R}^{M\times C}$, where $\mathbf{1} \in \mathbb{R}^{M}$ is a column one-vector, and then element-wise product with the value $\boldsymbol{X}_n^{\mathrm{K}}$. The resultant matrix, $\mathbf{1}(\boldsymbol{x}_n^{\mathrm{Q}})^{\mathsf{T}} \odot \boldsymbol{X}_n^{\mathrm{K}} \in \mathbb{R}^{M\times C}$, contains full element-wise correlations between the query (target) and the value (context). We generate the relational kernel $\boldsymbol{\kappa}_n^{\mathrm{R}}$ by projecting the vectorized correlations using a learnable matrix $\boldsymbol{H} \in \mathbb{R}^{MC\times M}$:

$$\boldsymbol{\kappa}_n^{\mathrm{R}} = \mathrm{vec}(\mathbf{1}(\boldsymbol{x}_n^{\mathrm{Q}})^{\mathsf{T}} \odot \boldsymbol{X}_n^{\mathrm{K}})\boldsymbol{H}, \tag{5}$$

where $\odot$ denotes the Hadamard product. The role of $\boldsymbol{H}$ in this relational kernel corresponds to that of $\boldsymbol{P}$ in the basic kernel of involution (Eq. 3); while $\boldsymbol{P}$ produces a kernel from C-dimensional target vector, $\boldsymbol{H}$ generates a kernel from $M \times C$ target-to-context correlations.

In training, $\boldsymbol{H}$ learns to predict the relevance of the context from target-context interactions, thus being particularly effective in learning motion-related patterns in videos; the resultant dynamic kernel aggregates the context depending on how the individual contents are related to each other in space and time. Furthermore, the absence of the softmax operator allows the relational kernel to have negative activations so that it better learns relative features where the subtractive interactions are beneficial for capturing relative changes of contents over time [44, 45, 46].

Relational context. The relational context is designed to provide the relational pattern of content-to-content (*i.e.*, context-to-context) interactions for the kernels to aggregate. To this end, we use self-correlation [49], which effectively describes spatio-temporal intra-structure including motion infomation [21, 22]. We first construct the self-correlation matrix $\boldsymbol{X}_n^{\mathrm{V}}(\boldsymbol{X}_n^{\mathrm{V}})^{\mathsf{T}} \in \mathbb{R}^{M\times M}$ and then project it using a learnable matrix $\boldsymbol{G} \in \mathbb{R}^{M\times C}$ into the relational context $\boldsymbol{X}_n^{\mathrm{R}}$:

$$\boldsymbol{X}_n^{\mathrm{R}} = \boldsymbol{X}_n^{\mathrm{V}}(\boldsymbol{X}_n^{\mathrm{V}})^{\mathsf{T}}\boldsymbol{G}, \tag{6}$$

where the self-correlation $\boldsymbol{X}_n^{\mathrm{V}}(\boldsymbol{X}_n^{\mathrm{V}})^{\mathsf{T}}$ reveals content-to-content interaction patterns within the context, and the matrix $\boldsymbol{G}$ maps it to the relational context so that the output has the same size with the basic context $\boldsymbol{X}_n^{\mathrm{V}}$.

4.2 Intervolution

Combining different types of kernels and contexts. The proposed transform, *intervolution*, integrates the relational kernel and context into a dynamic transform. As illustrated in Figure 2, it consists of two types of kernels, $\boldsymbol{\kappa}_n^{\mathrm{V}}$ and $\boldsymbol{\kappa}_n^{\mathrm{R}}$, and two types of contexts, $\boldsymbol{X}_n^{\mathrm{V}}$ and $\boldsymbol{X}_n^{\mathrm{R}}$. Note that the basic kernel $\boldsymbol{\kappa}_n^{\mathrm{V}}$ is computed as $\boldsymbol{x}_n^{\mathrm{Q}}\boldsymbol{P}^{\mathsf{T}}$. We combine the kernels and the contexts in the intervolution transform:

$$\begin{aligned}\boldsymbol{y}_n &= (\boldsymbol{\kappa}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{R}})(\boldsymbol{X}_n^{\mathrm{V}} + \boldsymbol{X}_n^{\mathrm{R}}) \\ &= \boldsymbol{\kappa}_n^{\mathrm{V}}\boldsymbol{X}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{R}}\boldsymbol{X}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{V}}\boldsymbol{X}_n^{\mathrm{R}} + \boldsymbol{\kappa}_n^{\mathrm{R}}\boldsymbol{X}_n^{\mathrm{R}},\end{aligned} \tag{7}$$

which contains four different dynamic transforms, $\boldsymbol{\kappa}_n^{\mathrm{V}}\boldsymbol{X}_n^{\mathrm{V}}$, $\boldsymbol{\kappa}_n^{\mathrm{R}}\boldsymbol{X}_n^{\mathrm{V}}$, $\boldsymbol{\kappa}_n^{\mathrm{V}}\boldsymbol{X}_n^{\mathrm{R}}$, and $\boldsymbol{\kappa}_n^{\mathrm{R}}\boldsymbol{X}_n^{\mathrm{R}}$. The first sub-transform, $\boldsymbol{\kappa}_n^{\mathrm{V}}\boldsymbol{X}_n^{\mathrm{V}}$ corresponds to the involution (Eq. 3). The second and third sub-transform, $\boldsymbol{\kappa}_n^{\mathrm{R}}\boldsymbol{X}_n^{\mathrm{V}}$ and $\boldsymbol{\kappa}_n^{\mathrm{V}}\boldsymbol{X}_n^{\mathrm{R}}$, both capture content-to-relation interactions, but their effects are different; while $\boldsymbol{\kappa}_n^{\mathrm{R}}\boldsymbol{X}_n^{\mathrm{V}}$ aggregates the basic context with considering the target-to-context interactions, $\boldsymbol{\kappa}_n^{\mathrm{V}}\boldsymbol{X}_n^{\mathrm{R}}$

Table 1: **Improving efficiency of intervolution.** N, M, C, D, H denotes the input size, context size, channel dimension, latent channel dimension, and number of queries, respectively.

Operation	Computational cost	Memory footprint
Intervolution	$\mathcal{O}(NM^2C)$	$\mathcal{O}(NM(M+C))$
+ Efficient $\boldsymbol{\kappa}_n^{\mathrm{R}}, \boldsymbol{X}_n^{\mathrm{R}}$	$\mathcal{O}(NMC(C+D))$	$\mathcal{O}(N(MC+C^2+D))$
+ Multi-query (H)	$\mathcal{O}(NMC(C+D)/H)$	$\mathcal{O}(N(MC+C^2+D)/H)$

aggregates the relational patterns with the content-to position interactions. The last sub-transform, $\boldsymbol{\kappa}_n^{\mathrm{R}}\boldsymbol{X}_n^{\mathrm{R}}$, is fully dependent on the relational information. The relational kernel dynamically aggregates relational patterns, generating the deeper relational interactions. These sub-transforms altogether encode rich relational interactions, leading to comprehensive video representation learning.

Improving efficiency of intervolution. We improve the efficiency by reducing the computational cost of relational kernel and relational context. The projection matrix $\boldsymbol{H} \in \mathbb{R}^{MC\times M}$ of relational kernel increases computation and memory quadratically with respect to the context size M, causing a computational bottleneck. We reduce the complexity to the linear one by decomposing $\boldsymbol{H}$ to $\boldsymbol{H}_1\boldsymbol{H}_2^{\mathsf{T}}$ such that $\boldsymbol{H}_1 \in \mathbb{R}^{MC\times D}$, $\boldsymbol{H}_2 \in \mathbb{R}^{M\times D}$, where D is a latent channel dimension that smaller than M. Our experiments show that decomposing $\boldsymbol{H}$ achieves a good computation-complexity tradeoff (please refer to Table. 4b). For reducing the computation of relational context, we first abstract the basic context to a contextual feature matrix $(\boldsymbol{X}_n^{\mathrm{V}})^{\mathsf{T}}\boldsymbol{G} \in \mathbb{R}^{C\times C}$ and then multiply it to $\boldsymbol{X}_n^{\mathrm{V}}$ for updating the basic context. By switching the orders in this manner, computation and memory can be reduced from quadratic to linear with respect to M as shown in Table 1. Note that C, D are channel dimensions that we can control. Furthermore, we adopt the multi-query setting [34] for intervolution, which set multiple queries and apply the same key and value context to each query. While the multi-head setting [5] maintains computational cost and increases memory, multi-query setting reduces computational cost and memory in proportion to the number of queries (Table 1).

5 Experiments

5.1 Implementation details

Action recognition architecture. We use TSN-ResNet50 [50] as our backbone and replace the standard spatial convolution layers by spatio-temporal intervolution layers for every two ResNet bottlenecks [27]. Unless specified otherwise, we replace 7 intervolution layers, there are 7 intervoluton layers in total where L=8, D=C, M=5×7×7. We set the input and output channel dimensions of intervolution layers to be equal to those of spatial convolution layers in TSN-ResNet50.

Training & testing details. For training, we use ImageNet-pre-trained weights for the initialization, except for the intervolution bottleneck blocks; we randomly initialize weights by using MSRA initialization [51] and set the gamma parameter of the last batch normalization layer to zero. We sample 8 or 16 frames from each video by using a segment-based random sampling strategy [50]. We resize the resolution of each frame to 240×320, apply random cropping, scale jittering, and random horizontal flipping for data augmentation. We use label smoothing [52], stochastic depth [53], and dropout for regularization. We set an initial learning rate to 0.02 and decay it by following a cosine learning rate schedule for 50 epochs including the initial 5 warm-up epochs, except, the models in Table 3a are trained for 80 epochs. We use SGD with a momentum of 0.9 and set the batch size as 64 across 8 V100 GPU machines. For testing, we densely sample one or two clips, resize them to 240 $\times$ 320, and center-crop as 224×224. We average softmax scores when ensemble multiple models. Please refer to the supplementary materials for the other implementation details.

5.2 Datasets

Something-something v1 & v2 (SS-v1 & v2) [25] are both large-scale action recognition benchmarks, including 108k and 220k action clips, respectively. Both datasets share the same motion-centric action classes, *e.g.*, 'pushing something from left to right,' so thus capturing fine-grained motion is crucial to achieve the better performance.

Table 2: **Performance comparison with other spatio-temporal feature transform methods on SS-v1**. $\boldsymbol{W}_j \circ \boldsymbol{W}_i(\cdot)$ indicates a sequential transform of $\boldsymbol{W}_i$ followed by $\boldsymbol{W}_j$. σ denotes softmax.

model (TSN-R50 [50])	kernel	context	FLOPs	params.	top-1	top-5
C2D	$\boldsymbol{W}_{\text{2D,standard}}$	$\boldsymbol{X}_n$	32.5 G	24.3 M	19.7	46.6
w/ C3D [15]	$\boldsymbol{W}_{\text{3D,standard}}$	$\boldsymbol{X}_n$	57.0 G	39.3 M	43.3	72.2
w/ R(2+1)D [17]	$\boldsymbol{W}_{\text{1D,standard}} \circ \boldsymbol{W}_{\text{2D,standard}}$	$\boldsymbol{X}_n$	37.3 G	26.8 M	44.1	72.9
w/ LSA [32]	$\sigma(\boldsymbol{x}_n^{\text{Q}}(\boldsymbol{X}_n^{\text{K}})^{\mathsf{T}} + \boldsymbol{x}_n^{\text{Q}}\boldsymbol{P}^{\mathsf{T}}))$	$\boldsymbol{X}_n^{\text{V}}$	32.2 G	23.4 M	41.6	70.9
w/ LSA variant [32]	$\sigma(\boldsymbol{x}_n^{\text{Q}}\boldsymbol{P}^{\mathsf{T}})$	$\boldsymbol{X}_n^{\text{V}}$	32.1 G	23.4 M	41.3	70.6
w/ LSA variant [32]	$\boldsymbol{x}_n^{\text{Q}}\boldsymbol{P}^{\mathsf{T}}$	$\boldsymbol{X}_n^{\text{V}}$	32.1 G	23.4 M	44.3	73.7
w/ Intervolution	$\boldsymbol{\kappa}_n^{\text{V}} + \boldsymbol{\kappa}_n^{\text{R}}$	$\boldsymbol{X}_n^{\text{V}}$	32.6 G	23.4 M	45.7	74.8
w/ Intervolution	$\boldsymbol{\kappa}_n^{\text{R}} + \boldsymbol{\kappa}_n^{\text{V}}$	$\boldsymbol{X}_n^{\text{V}} + \boldsymbol{X}_n^{\text{R}}$	32.6 G	23.6 M	**47.0**	**75.7**

Diving-48 [54] is a fine-grained action benchmark that is heavily dependent on temporal modeling [20], containing 18k videos with 48 diving classes. Due to the incorrect label issue, we only compare our result with the results on the modified version of Diving-48.

FineGym [55] is a motion-centric benchmark that includes gymnastics action classes. We report results on two subsets of *Gym288* and *Gym99* that contain 288 and 99 action classes, respectively.

5.3 Comparison with other transform methods.

In this experiment, we evaluate the temporal modeling capability of different transform methods: spatio-temporal convolutions [15, 17, 35], local self-attention (LSA) [32], LSA variants, and intervolution. We replace a single 3×3 spatial convolution layer in TSN-ResNet50 [50] with a single spatio-temporal transform layer. We analyze the ability of modeling temporal dependency of each transform layer with an apple-to-apple comparison. We use 8 frames as the input, and the kernel size M of all spatio-temporal transforms is set to $5\times7\times7$.

Table 2 summarizes the results. Spatial convolution baseline (C2D) without modeling temporal dependency shows the lowest accuracy of 19.7%. While C3D [15] and R(2+1)D [17], which use static spatio-temporal transforms, show the top-1 accuracy of 43.3% and 44.1%, respectively, LSA only achieves 41.6%. To find out the reasons behind the bad result, we ablate each component of LSA, *e.g.*, query-key correlation and softmax, one by one. The LSA variant without query-key correlation achieves comparable performance to LSA. It indicates that the temporal modeling capability of LSA is actually dependent on content-to-position correlation rather than content-to-content correlation; the self-attention mechanism itself is permutation-invariant, so thus LSA without positional embedding is hard to learn position-specific features, *e.g.*, motion [24]. After we remove the softmax non-linearity ($\boldsymbol{x}_n^{\text{Q}}\boldsymbol{P}^{\mathsf{T}}\boldsymbol{X}_n^{\text{V}}$), where the kernel is equivalent to the basic kernel ($\boldsymbol{\kappa}_n^{\text{V}} = \boldsymbol{x}_n^{\text{Q}}\boldsymbol{P}^{\mathsf{T}}$), outperforms the both LSA and the standard static transforms. It demonstrates the softmax function restricts the expressive ability of the kernel [9]; the softmax forces the kernel weights to be positive, so that the kernel may not compute gradients across frames, which are effective in learning motions. Please refer to the supplementary materials for the qualitative results.

While the composability of the involution kernel depends on the target content, our relational kernel depends on the local pattern of target-context correlation. As we add the relational kernel $\boldsymbol{\kappa}_n^{\text{R}}$ to $\boldsymbol{\kappa}_n^{\text{V}}$, we improve the top-1 accuracy by 1.4%p. The result demonstrates that leveraging local target-context patterns are effective in aggregating informative context for learning motions. At last, by adding relational context $\boldsymbol{X}_n^{\text{R}}$ to basic context $\boldsymbol{X}_n^{\text{V}}$, intervolution achieves the best top-1 accuracy of 47.0%.

5.4 Comparison to the state-of-the-art methods

Table 3a compares our method to the state-of-the-art methods on SS-v1 and v2 datasets. The first compartment shows the results of the 3D CNNs. The second compartment contains the motion-modeling methods: both STM [45] and TEA [46] compute frame-wise differences, and both CorrNet [21] and MSNet [22] compute inter-frame pixel-wise correlations to learn motion features. The third compartment reports the results of global self-attention-based models. NL-I3D [23] inserts non-local blocks to the 3D CNN for capturing long-range spatio-temporal dependencies. TimeSformer-L,

Table 3: **Performance comparison on SS-V1&V2, FineGym, and Diving-48**.

(a) **SS-V1&V2**. IN and IN21K and K400 denote ImageNet-1k, ImageNet-21K, and Kinetics-400 dataset, respectively. Our method achieves a new state-of-the-art accuracy on both datasets.

model	pre-trained	#frame	FLOPs ×clips	SS-V1 top-1	SS-V1 top-5	SS-V2 top-1	SS-V2 top-5
I3D [16] from [56]	IN	32	153 G×2	41.6	72.2	-	-
TSM-R50 [37]	IN	16	65 G×1	47.2	77.1	63.4	88.5
ir-CSN-152 [35]	-	32	97 G×10	49.3	-	-	-
SlowFast8×8-R50 [57]	K400	32	67 G×3	-	-	61.7	86.9
CT-Net-R50 [58]	IN	16	75 G × 1	52.5	80.9	64.5	89.3
STM-R50 [45]	IN	16	67 G×30	50.7	80.4	64.2	89.8
CorrNet-R101 [21]	-	32	187 G×10	50.9	-	-	-
TEA [46]	IN	16	70 G ×3	52.3	81.9	65.1	89.9
MSNet-TSM-R50 [22]	IN	16	67 G×1	52.1	82.3	64.7	89.4
NL-I3D [23] from [56]	IN	32	168 G×2	44.4	76.0	-	-
TimeSformer-HR [20]	IN	16	1703 G×3	-	-	62.2	-
TimeSformer-L [20]	IN	96	2380 G×3	-	-	62.4	-
ViViT-L [18]	IN21K & K400	32	N/A×4	-	-	65.4	89.8
Intervolution-R50 (ours)	IN	8	36 G×1	51.9	79.6	64.8	89.1
Intervolution-R50 (ours)	IN	16	72 G×1	54.0	81.1	66.0	89.8
Intervolution-R50$_{EN}$ (ours)	IN	8+16	108 G×1	55.5	82.6	67.3	90.8
Intervolution-R50$_{EN}$ (ours)	IN	8+16	108 G×2	**56.1**	**82.8**	**67.7**	**91.1**

(b) **Diving-48**. Top-1 accuracy, FLOPs are shown. Results in the upper compartment are from [20].

model	FLOPs ×clips	top-1
SlowFast-R101 [57]	213 G×3	77.6
TimeSformer [20]	196 G×3	75.0
TimeSformer-HR [20]	1703 G×3	78.0
TimeSformer-L [20]	2380 G×3	81.0
Intervolution-R50	72G×2	**84.2**

(c) **FineGym**. The averaged per-class accuracy (%) is reported. All results in the upper compartment are from [55].

model	Gym288	Gym99
TRN [59]	33.1	68.7
I3D [16]	27.9	63.2
TSM [37]	34.8	70.6
TSM$_{\text{Two-stream}}$ [37]	46.5	81.2
Intervolution-R50	**50.9**	**86.4**

TimeSformer-HR [20] and ViViT [18] are transformer architectures that learn video representations via factorized spatio-temporal self-attention. Our method, Intervolution-R50, achieves 51.9% and 64.8% at top-1 accuracy on SS-v1 and v2 datasets, respectively, which are already competitive to most of existing methods while using 8 frames only. When we use 16 frames, our method outperforms other existing models by achieving 54.0% and 66.0% at top-1 accuracy on SS-v1 and v2, respectively. Finally, our ensemble model with 2 clips achieves 56.1% and 67.7%, which set the new state-of-the-art on SS-v1 and v2 with reasonable computational cost.

Table 3c and Table 3b present the results on Diving-48 [54] and FineGym [55]. For Diving-48, our model achieves 84.2%, substantially outperforming the state-of-the-art 3D CNN and transformer architectures. For FineGym, our model outperforms other methods in the averaged per-class accuracy of 50.9% and 86.4% given 288 and 99 classes, respectively.

5.5 Ablation studies

We conduct ablation experiments to validate the effectiveness of intervolution. We use 8 8frames for all experiments. Other training and testing details are in Sec. 5.1.

Combinations of different kernels and contexts. In Table 4a, we compare the performance of a single intervolution layer with different combinations of dynamic kernels and contexts. We fix the context as $\boldsymbol{X}_n^{\mathrm{V}}$ and then vary different types of the kernels from $\boldsymbol{\kappa}_n^{\mathrm{V}}$, $\boldsymbol{\kappa}_n^{\mathrm{R}}$ to $\boldsymbol{\kappa}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{R}}$, while the context is fixed as $\boldsymbol{X}_n^{\mathrm{V}}$. Compared to $\boldsymbol{\kappa}_n^{\mathrm{V}}\boldsymbol{X}_n^{\mathrm{V}}$, $\boldsymbol{\kappa}_n^{\mathrm{R}}\boldsymbol{X}_n^{\mathrm{V}}$ improves the accuracy by 0.6%. It indicates that predicting kernel from the composition of local target-context correlation is effective in modeling temporal dependencies. As we use both of the basic and relational transforms, $(\boldsymbol{\kappa}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{R}})\boldsymbol{X}_n^{\mathrm{V}}$, we obtain additional improvements by 0.3%p and 0.9%p at top-1 and top-5 accuracy, respectively. We also vary different types of the context such as $\boldsymbol{X}_n^{\mathrm{V}}$, $\boldsymbol{X}_n^{\mathrm{R}}$, and $\boldsymbol{X}_n^{\mathrm{V}} + \boldsymbol{X}_n^{\mathrm{R}}$, while the kernel is fixed. We observe the consistent improvements across different types of contexts, which means that aggregating the relational context are beneficial for learning relational information. Finally, as we combine all dynamic transforms, $(\boldsymbol{\kappa}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{R}})(\boldsymbol{X}_n^{\mathrm{V}} + \boldsymbol{X}_n^{\mathrm{R}})$, we achieve the highest accuracy of 47.0%.

Latent dimension D. In Table 4b, we validate the effectiveness of decomposing $\boldsymbol{H}$ with latent channel dimension D. Without decomposing $\boldsymbol{H}$, the TSN-ResNet50 with 7 intervolution layers requires 62.9 GFLOPs and 32.0 M parameters, resulting in out-of-memory error. After we decompose $\boldsymbol{H}$ with a small D, we reduce FLOPs by about a half. We set $D = C$ as the default that performs the highest accuracy.

Table 4: **Ablation studies on SS-v1 dataset**. All models use TSN-ResNet50 [50] as the backbone. Top-1, top-5 accuracy (%), FLOPs (G) and paramaters (M) are shown.

(a) **Combinations of different kernels and context**. A single intervolution layer is inserted into stage4.

kernel	context	FLOPs	params.	top-1	top-5
$\boldsymbol{\kappa}_n^{\mathrm{V}}$	$\boldsymbol{X}_n^{\mathrm{V}}$	32.3 G	23.4 M	44.8	73.8
$\boldsymbol{\kappa}_n^{\mathrm{R}}$	$\boldsymbol{X}_n^{\mathrm{V}}$	32.7 G	23.6 M	45.4	73.9
$\boldsymbol{\kappa}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{R}}$	$\boldsymbol{X}_n^{\mathrm{V}}$	32.7 G	23.6 M	45.7	74.8
$\boldsymbol{\kappa}_n^{\mathrm{V}}$	$\boldsymbol{X}_n^{\mathrm{R}}$	32.7 G	23.4 M	46.2	75.4
$\boldsymbol{\kappa}_n^{\mathrm{R}}$	$\boldsymbol{X}_n^{\mathrm{R}}$	33.2 G	23.6 M	46.5	75.6
$\boldsymbol{\kappa}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{R}}$	$\boldsymbol{X}_n^{\mathrm{R}}$	33.2 G	23.6 M	46.7	75.6
$\boldsymbol{\kappa}_n^{\mathrm{V}}$	$\boldsymbol{X}_n^{\mathrm{V}}+\boldsymbol{X}_n^{\mathrm{R}}$	32.7 G	23.4 M	46.5	75.6
$\boldsymbol{\kappa}_n^{\mathrm{R}}$	$\boldsymbol{X}_n^{\mathrm{V}}+\boldsymbol{X}_n^{\mathrm{R}}$	33.2 G	23.6 M	46.8	75.6
$\boldsymbol{\kappa}_n^{\mathrm{V}} + \boldsymbol{\kappa}_n^{\mathrm{R}}$	$\boldsymbol{X}_n^{\mathrm{V}}+\boldsymbol{X}_n^{\mathrm{R}}$	33.2 G	23.6 M	**47.0**	**75.7**

(b) **Latent dimension** D. Decomposing $\boldsymbol{H}$ significantly reduces the computation cost. OOM is an abbrebiation of out-of-memory. 8 video clips per a single GPU machine are used.

D	FLOPs	params.	memory	top-1	top-5
-	62.9 G	32.0 M	OOM	OOM	OOM
8	32.2 G	20.5 M	8.8 GB	50.1	78.8
16	34.7 G	20.9 M	9.2 GB	51.3	78.8
32	39.6 G	21.7 M	10.2 GB	50.9	79.0
$C/2$	32.9 G	21.1 M	8.8 GB	51.1	79.1
C	35.9 G	22.0 M	9.6 GB	**51.5**	**79.2**

(c) **Kernel size** M. In most cases, larger kernel results in the higher accuracy.

kernel size M	FLOPs	params.	top-1	top-5
$3 \times 3 \times 3$	28.5 G	20.3 M	49.4	77.6
$3 \times 5 \times 5$	30.2 G	20.7 M	50.5	78.7
$3 \times 7 \times 7$	32.6 G	21.2 M	50.7	78.9
$3 \times 9 \times 9$	35.8 G	22.0 M	51.1	79.1
$5 \times 7 \times 7$	35.9 G	22.0 M	**51.5**	**79.2**
$5 \times 9 \times 9$	41.3 G	23.3 M	51.2	78.9

(d) **Number and locations of Intervolution layers**. More intervolution layers lead the better performance.

number	positions	FLOPs	params.	top-1	top-5
0	-	32.5 G	24.3 M	19.7	46.6
1	stage4	33.2 G	23.6 M	47.0	75.7
2	stage4-5	33.7 G	22.6 M	47.2	76.1
4	stage4-5	34.6 G	22.2 M	50.4	78.7
7	stage2-5	35.9 G	22.0 M	**51.5**	**79.2**

(e) **Normalization**. L2-normalization performs the best accuracy in our setting.

normalization	FLOPs	params.	top-1	top-5
no norm.	35.9 G	22.0 M	50.3	78.6
batch norm.	35.9 G	22.0 M	49.4	78.0
norm. from [34]	35.8 G	22.0 M	49.6	78.2
L2-norm.	35.9 G	22.0 M	**51.5**	**79.2**

Kernel sizes. In Table 4c, we compare the effect of the kernel size M. In most cases, larger spatio-temporal kernel improves the accuracy, except the case of $M = 5 \times 9 \times 9$. Considering the static convolution counterparts, the intervolution effectively enlarges the spatio-temporal kernel size, requiring smaller FLOPs and parameters. We choose $M = 5 \times 7 \times 7$ as the default kernel size that shows the best computation-accuracy trade-off.

Number and locations of Intervolution layers. In Table 4d, we gradually replace the spatial convolutional layers with intervolution layers. Adding a single intervolution layer significantly improves the top-1 accuracy by 27.3%p. As we increase the number of intervolution layers, we obtain gradual improvements of accuracy, while reducing parameters only. Note that intervolution layer with smaller kernel size M, *e.g.*, $3 \times 5 \times 5$, requires smaller amount of both FLOPs and parameters, while covering larger receptive fields than the convolution as shown in the second row of Table 4c.

Normalization. In Table 4e, we investigate different normalization methods to $\boldsymbol{x}_n^{\mathrm{Q}}$, $\boldsymbol{X}_n^{\mathrm{K}}$, and $\boldsymbol{X}_n^{\mathrm{V}}$. In our setting, applying L2 normalization performs the best. Notably, applying no normalization perform better than applying batch normalization or normalization scheme from [34] due to the ImageNet-pre-trained weights.

6 Conclusion

We have presented the intervolution feature transform, which captures rich relational patterns for video understanding, and validated that it outperforms other dynamic feature transforms in learning motion dynamics in videos. The proposed intervolution networks outperform the state-of-the art methods on standard motion-centric action benchmarks.

While we have focused on video representation learning in this work, we believe the intervolution will also benefit image understanding and natural language processing. We leave this for future work.

References

[1] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*. 1982.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[3] 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*, pages 3464–3473, 2019.

[4] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *NIPS*, 2016.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2017.

[6] Ningning Ma, Xiangyu Zhang, Jiawei Huang, and Jian Sun. Weightnet: Revisiting the design space of weight networks. *arXiv preprint arXiv:2007.11823*, 2020.

[7] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020.

[8] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *arXiv preprint arXiv:1904.04971*, 2019.

[9] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. *arXiv preprint arXiv:2103.06255*, 2021.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[12] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *arXiv preprint arXiv:2103.12731*, 2021.

[13] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.

[14] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. *arXiv preprint arXiv:2105.05633*, 2021.

[15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015.

[16] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021.

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

[20] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021.

[21] Heng Wang, Du Tran, Lorenzo Torresani, and Matt Feiszli. Video modeling with correlation networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[22] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Motionsqueeze: Neural motion feature learning for video understanding. *arXiv preprint arXiv:2007.09933*, 2020.

[23] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[24] Xingyu Liu, Joon-Young Lee, and Hailin Jin. Learning video representations from correspondence proposals. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] 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 *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[29] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. International Conference on Machine Learning (ICML)*, 2019.

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[31] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

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

[33] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020.

[34] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021.

[35] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019.

[36] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] Ji Lin, Chuang Gan, and Song Han. Tsm: Temporal shift module for efficient video understanding. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019.

[38] Linxi Fan, Shyamal Buch, Guanzhi Wang, Ryan Cao, Yuke Zhu, Juan Carlos Niebles, and Li Fei-Fei. Rubiksnet: Learnable 3d-shift for efficient video action recognition. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.

[39] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. Neural Information Processing Systems (NeurIPS)*, 2014.

[40] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[41] Lijie Fan, Wenbing Huang, Chuang Gan, Stefano Ermon, Boqing Gong, and Junzhou Huang. End-to-end learning of motion representation for video understanding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[42] AJ Piergiovanni and Michael S Ryoo. Representation flow for action recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[43] 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 *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[44] Myunggi Lee, Seungeui Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

[45] Boyuan Jiang, Mengmeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019.

[46] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[47] Heeseung Kwon, Manjin Kim, Suha Kwak, and Minsu Cho. Learning self-similarity in space and time as generalized motion for action recognition. *arXiv preprint arXiv:2102.07092*, 2021.

[48] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[49] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[50] 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 *Proc. European Conference on Computer Vision (ECCV)*, 2016.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[53] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

[54] Yingwei Li, Yi Li, and Nuno Vasconcelos. Resound: Towards action recognition without representation bias. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

[55] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Finegym: A hierarchical video dataset for fine-grained action understanding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[56] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proc. European Conference on Computer Vision (ECCV)*, pages 399–417, 2018.

[57] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019.

[58] Kunchang Li, Xianhang Li, Yali Wang, Jun Wang, and Yu Qiao. {CT}-net: Channel tensorization network for video classification. In *International Conference on Learning Representations*, 2021.

[59] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proc. European Conference on Computer Vision (ECCV)*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Our abstract and introduction accurately reflects the contributions of the manuscript.
   (b) Did you describe the limitations of your work? [Yes] We will include the further improvements of our work in supplementary materials.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Our paper conforms to the ethics review guidelines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will include the configurations and details in supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We will specify all the training details in supplementary materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Unfortunately, we conducted main experiments once due to the lack of GPU resources against the scale of the task (video action recognition). Instead, we are going to submit the detailed training logs of our main experiments in supplementary materials.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We include the total amount of compute and the type of the GPUs in Sec. 5.1.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited all the benchmarks we used.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]