Selective Dependency Aggregation for Action Classification

Yi Tan[1], Yanbin Hao[1§], Xiangnan He[1], Yinwei Wei[2], Xun Yang[2]
[1]University of Science and Technology of China, [2]National University of Singapore
ty133@mail.ustc.edu.cn,{haoyanbin,xweiyinwei}@hotmail.com,{xiangnanhe,hfutyangxun}@gmail.com

ABSTRACT

Video data are distinct from images for the extra temporal dimension, which results in more content dependencies from various perspectives (i.e., long-range and short-range). It increases the difficulty of learning representation for various video actions. Existing methods mainly focus on the dependency under a specific perspective, which cannot facilitate the categorization of complex video actions. This paper proposes a novel **s**elective **d**ependency **a**ggregation (SDA) module, which adaptively exploits multiple types of video dependencies to refine the features. Specifically, we empirically investigate various long-range and short-range dependencies achieved by the multi-direction multi-scale feature squeeze and the dependency excitation. Query structured attention is then adopted to fuse them selectively, fully considering the diversity of videos' dependency preferences. Moreover, the channel reduction mechanism is involved in SDA for controlling the additional computation cost to be lightweight. Finally, we show that the SDA module can be easily plugged into different backbones to form SDA-Nets and demonstrate its effectiveness, efficiency and robustness by conducting extensive experiments on several video benchmarks for action classification. The code and models will be available at https://github.com/ty-97/SDA.

CCS CONCEPTS

• **Computing methodologies → Activity recognition and understanding**.

KEYWORDS

Action classification; Video content dependency; Selective dependency aggregation

ACM Reference Format:
Yi Tan[1], Yanbin Hao[1§], Xiangnan He[1], Yinwei Wei[2], Xun Yang[2]. 2021. Selective Dependency Aggregation for Action Classification. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3474085.3475218

1 INTRODUCTION

Capturing content-related dependencies is of central importance in such as natural language processing (NLP) [41], image processing [4], and the studied action classification [43]. Unlike the processing on one and two dimensional signals, i.e., the language sequence and the static image, modeling and utilizing dependencies on 3D video signals are more challenging in action classification and other downstream tasks such as video retrieval [48, 49] and content analysis [1, 29, 35, 50, 51]. The difficulties mainly lie in two aspects. First, the 3D dynamic nature inherently widens the sphere of actions with an order of magnitude larger than the 2D static vision, resulting in multiple dependencies across space and time dimensions. While in contrast, the language/image data mostly exhibit long-rang/-distance modeling. However, existing dependency modeling methods mainly leverage the dependency under a specific view, such as temporal (i.e. TEA [24] and TPN [47]) and global (i.e. S3D-G [46] and non-local network [43]) perspectives. Consequently, how to organize those various spatio-temporal dependencies based on video contents is a key problem. Second, video neural network models (e.g., C3D [38] and I3D [5]) commonly contain much more parameters and are hard to train. Further deepening the model (e.g., I3D [5]) or adding pairwise spatio-temporal attentions (e.g., Non-local network [43]) to capture long-range dependency will additionally incur significant computational burden. How to significantly reduce the extra computational cost is another key consideration.

Dependencies between video contents reflect the relationships among the 3D spatio-temporal variations, which can be long-range (i.e., the whole video) and short-range (i.e., local part of the video) in space, time and space-time. Generally, those multi-dependencies contribute unequally to action classification, since the semantic categories in different videos rely on different content interactions. For example, in Figure 1, the diving video needs to aggregate the

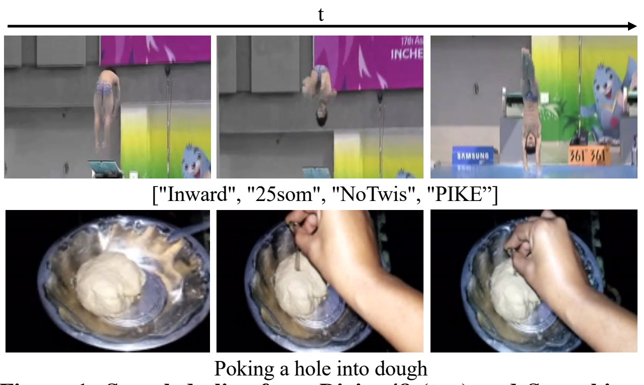

Figure 1: Sampled clips from Diving48 (top) and Something-Something (bottom). To understand the dive ["Inward", "25som", "NoTwis", "PIKE"], aggregating sub-poses in the dive sequences, which can be regarded as long-range temporal dependency modeling, is crucial. As for recognizing "Poking a hole into dough", Long-range temporal dependency and short-range spatial interactions are needed.

§ Yanbin Hao is the corresponding author.

sub-poses in dive sequences along the timeline for action modeling. Whereas, the recognition of the action "Poking a hole into a dough" requires both short-range spatial (i.e., the interactions between objects of *dough*, *spoon* and *hand*), and long-range temporal (i.e., motion dependencies over the whole time) modeling. These examples show clues that different videos may rely on different content-related dependencies, and properly capturing those dependencies can benefit the video categorization.

Towards the aforementioned challenges in action classification, we propose the **selective dependency aggregation** (SDA) module, an efficient and effective plug-and-play module for extracting and organizing multiple content-related dependencies with a low computation cost. Specifically, it mainly consists of a multi-dependency modeling (MDM) block and a dependency aggregation (DAG) block, where the MDM block is designed for modeling various space-time dependencies from input video features and the DAG block is for aggregating these dependencies. It is worth noting that before the dependency modeling, we initially perform channel reduction for memory and computation efficiency.

In MDM, we propose to squeeze a given 3D spatio-temporal feature along different directions and with multiple scales to obtain multiple spatio-temporal dependencies (e.g., long-range, short-range). As described in the squeeze-and-excitation network (SE-Net) [19], the dependency feature here is referred to as the information aggregated from a specific receptive field of the input feature. For example, when shrinking the feature across space and time, the dependency feature is a vector where the global spatial-temporal content is stored [46]. As a contrast, if we pool the feature using a kernel with small size, elements in the dependency feature cube thus represent local relations. Through changing the receptive filed, we can obtain multiple dependency features. To achieve the goal of efficient modeling, we build our dependency modeling operation as a simple and general cascaded structure of "feature-squeeze $\rightarrow$ dependency-excitation". The "feature-squeeze" operation (e.g., pooling layer) is for providing the dependency feature and the " dependency-excitation" operation (e.g. FC layer or convolution layer followed by an activation function) is for modeling the feature-level dependencies from the squeezed features. Hence, various dependencies are modeled fully and properly.

In DAG, considering the fact that different videos exhibit different dependencies, the ability of dynamic selection for multiple dependencies is needed for better action understanding. To this end, we use the query structured attention (QSA) [26] to adaptively assign weights to different dependencies and combine them with a weighted sum of dependency responses. The QSA changes the "query" in transformer [41] to a learnable vector and treats the input itself as "key" and "value" for computation efficiency. Finally, the combined dependency representation is projected to a tensor of the same size as the original input video feature, and regularized through the Sigmoid function to produce a collection of element-wise modulation gating weights. These weights are applied to reweight the input feature by element-wise production. The SDA module is densely inserted into each residual block of the existing video networks to achieve layer-wise feature refinement.

We summarize our contributions as below:

- We propose to model multiple dependencies, including various long-range and short-range variants, facilitating the feature refinement of video features.
- We construct a dependency aggregation block, where the QSA method is adopted to dynamically assign attention weights to those dependencies. So that the most helpful dependency can be emphasized with a higher weight according to the video contents.
- Our proposed SDA module is a plug-and-play unit and can be conveniently inserted into the off-the-shelf action classification models such as TSN [42] and TSM [27] without incurring much overhead (i.e., 7.9%/3.0% extra parameters/FLOPs). Moreover, experimental results on four benchmarks, including Something-Something V1&V2, Diving48, and EPIC-KITCHEN-55 datasets, show the effectiveness of our method.

2 RELATED WORK

Deep video networks. As deep convolution networks have brought great progress for static visual content modeling, various CNN-based deep video architectures have been proposed to handle the classification of video data.

The most classical works [9, 18, 21, 42] directly extend the successful 2D CNNs for video recognition. Here, 2D convolutions are simply employed to model static visual contents from separate frames in different layers. Then, they fuse the extracted features across frames to achieve temporal modeling. For example, Karpathy *et al.* [21] attempt to averagely pool the frame-level CNN features at different stages (e.g., early and late) for the clip-level result. Yue-Hei *et al.* [52] input the frame features extracted from 2D CNN into the recurrent neural network (RNN), e.g., LSTM [18], to organize the temporal orders. Donahue *et al.* [9] further explore to train the "CNN+RNN" model in an end-to-end fashion. Later, temporal segment network (TSN) [42] proposes to fuse the per-frame prediction scores with a segmental consensus function. Temporal relation network (TRN) [54] replaces the pooling operation of TSN with multi-layer perceptrons. Although these networks are computational friendly, they mainly focus on spatial modeling and hence perform less satisfactorily on videos requiring more temporal modeling.

Current models [5, 38] propose to design 3D spatio-temporal units to jointly process spatial and temporal signals in each layer. The most general spatio-temporal unit is the 3D convolution. Specifically, C3D [38] simply expands the kernel slides of the 2D sptial convolutions in such as ResNets [17] to 3 dimensions (i.e., space and time). I3D [5] initializes the 3D convolutions by inflating the 2D convolutions pretrained on ImageNet [34] to benefit the deeper model training. Moreover, V4D [53] even adopts 4D convolution to additionally capture the interactions among sub-clips. As the 3D/4D convolutions incur a huge number of parameters, research efforts [12, 33, 40, 46] have shifted to seek lightweight alternatives for the above heavy computational units. For example, P3D [33], R(2+1)D [40] and S3D [46] decompose the 3D convolution into the cascade of 2D spatial convolution and 1D temporal convolution, resulting in significant complexity reduction and performance improvement. SlowFast [12] further introduces two CNN paths to operate on different sampling frequencies and focuses on only the temporal relations (i.e., slow and fast movements) without considering much on the spatial dependencies. In addition, GST [31] and CSN [39] use group convolutions to separate channel interactions and spatio-temporal interactions, achieving much more efficient models. X3D [11] expands a tiny 2D CNN to facilitate processing 3D signals by stepwise searching for optimal settings for space, time, width and depth.

Except the above pure convolution-based spatio-temporal units, there are also some shift-based units that are almost parameter-free. TSM [27] replaces the 1D temporal convolution in such as R(2+1)D with the temporal shift of partial channels, where its effectiveness has been demonstrated in both action recognition and detection scenarios [15]. GSM [37] extends TSM with learnable shift parameters and uses the channel decomposition to further reduce parameters. Moreover, RubikShift [10] even tries to replace all convolutional filters with lightweight spatial/temporal shift operations.

Dependency modeling. Dependency is of crucial importance in signal processing, scanning from natural language processing(NLP), image processing, to video understanding.

To aggregate long-range dependency between various corpus, RNN-base models, long short-term memory (LSTM) [18] and gated recurrent [7] neural networks in particular, are proposed. With the inherent ability of remembering the long sequence, these methods succeed in modeling long-range dependency for NLP tasks such as language modeling and machine translation [2, 6]. However, RNN-based models fail to encode sequential data in parallel, and hence require more time to train. To overcome the limitation, Self-Attention [41] which relies entirely on an attention mechanism to draw global dependencies and allows for significantly more parallelization, has been introduced to compute representations for long-length sequences and achieves very promising results on various NLP tasks.

chuAs for image processing, traditional architectures grip the short-range dependency by sliding convolutional kernel with limited receptive field and the long-range dependency by stacking convolutions in an implicit manner [13, 22]. Later researchers design extra dependency modeling units to tackle this problem explicitly. SE-Net [19] is proposed to refine the learnt image feature layer by layer through plugging in a squeeze and excitation module upon the global dependency aggregated by global average pooling. CBAM [45] additionally considers the content-based dependency from the spatial perspective. Facing more complex dependencies between video contents, the non-local operator [43] models long-range dependency by computing a neural response of the local receptive field as a weighted sum of features across all spatio-temporal positions on the 3D feature map. As two sides of a coin, the non-local network is effective but suffers from heavy computation due to the pairwise distance calculation across a spatio-temporal feature cube. Besides, to model the long-range spatio-temporal dependency efficiently, S3D-G [46] aggregates long-range dependency through squeezing global spatio-temporal contexts along the channel dimension and refine the learnt feature of S3D under the long-range context dependency in a self-gating attention manner. TEA [24] extends SE-Net [19] proposed for image processing to enhance models with aggregated temporal context, while TPN [47] boosts the TSM [27] by aggregating the information of various visual tempos at the feature level.

These methods only view the complex dependencies from a specific perspective, leaving the problem of simultaneously modeling of multiple dependencies unexplored.

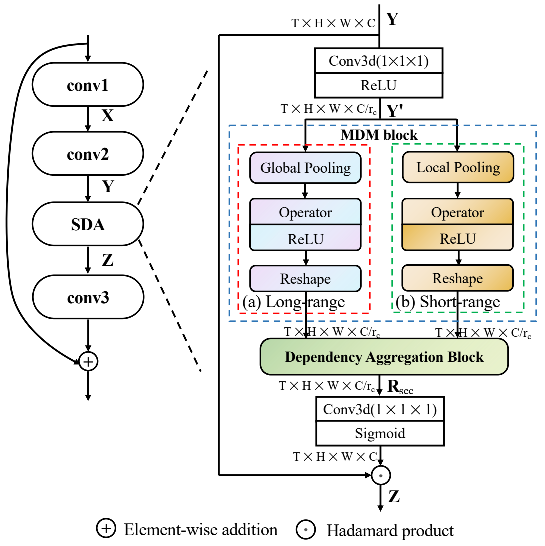

Figure 2: Framework of the proposed SDA module.

3 SELECTIVE DEPENDENCY AGGREGATION

Our selective dependency aggregation (SDA) module consists of two blocks, i.e., the multi-dependency modeling (MDM) block and the dependency aggregation (DAG) block. Below, we elaborate on the designing of the two blocks and also give an analysis to model complexity for SDA. Generally, we integrate the SDA unit to a residual block of ResNets. Figure 2 illustrates the framework of the proposed SDA.

3.1 Multi-dependency Modeling

The MDM block obtains multiple space-time dependencies from a given video feature tensor $\mathbf{Y} \in \mathbb{R}^{T \times H \times W \times C}$ outputted by such as a convolution layer. To avoid introducing much computational burden, a convolution layer followed by a ReLU activation function, particularly, is used to reduce the dimensions of channel C controlled by a hyperparameter r_c, yielding a new feature tensor $\mathbf{Y}' \in \mathbb{R}^{T \times H \times W \times \frac{C}{r_c}}$. As the proposed MDM block can work upon any given 4D video feature, we present it in a general fashion.

Given $\mathbf{Y}'$ as input, MDM outputs a set of dependency representations $\{\boldsymbol{R}_1, \boldsymbol{R}_2, \cdots, \boldsymbol{R}_M\}$, where M denotes the number of considered dependencies. Formally, we have

$$\{\boldsymbol{R}_1, \boldsymbol{R}_2, \cdots, \boldsymbol{R}_M\} = \mathrm{MDM}\left(\mathbf{Y}'\right). \tag{1}$$

The calculations of different dependency representations share a similar pipeline of "feature-squeeze → dependency-excitation". Specifically, we instantiate the "feature-squeeze" with an average pooling operation. Considering the space-time attribute of $\mathbf{Y}'$, we can pool it along different directions (e.g., space dimension, time dimension) and use different scales, referred to as multi-direction multi-scale squeeze. For notation clarity, we use the pooling kernel $W^{Pool}_{p_t,p_h,p_w} = (p_t, p_h, p_w)$, where p_t, p_h, p_w denote the size of the receptive field, to specify the average pooling operations. For example, if we set $(p_t, p_h, p_w) = (T, H, W)$, i.e., using the kernel $W^{Pool}_{T,H,W}$, we can obtain a $\frac{C}{r_c}$ vector that represents the global information squeezed across space and time. By applying the average pooling operation denoted as $Pool_{avg}$ over the input tensor $\mathbf{Y}'$, the squeezed feature $\mathbf{A} \in \mathbb{R}^{\frac{T}{p_t} \times \frac{H}{p_h} \times \frac{W}{p_w} \times \frac{C}{r_c}}$ can be computed as follows

$$\mathbf{A} = Pool_{avg}\left(\mathbf{Y}'; W^{Pool}_{p_t,p_h,p_w}\right). \tag{2}$$

The squeezed feature provides statistical information squeezed in a receptive field for dependency modeling. After obtaining the dependency feature $\mathbf{A}$, the task is reduced to how to excite the dependency feature and form the dependency representation, i.e., "dependency-excitation". Here, we use a convolution-based operation to achieve the dependency excitation. Similar to $Pool_{avg}$, the convolution function $Conv3d$ is also specified by the kernel $W^{Conv}_{c_t,c_h,c_w} = (c_t, c_h, c_w)$. As a result, given the dependency feature $\mathbf{A}$ as input, we can compute the corresponding dependency representation $\mathbf{R} \in \mathbb{R}^{\frac{T}{p_t} \times \frac{H}{p_h} \times \frac{W}{p_w} \times \frac{C}{r_c}}$ as

$$\mathbf{R} = ReLU\left(Conv3d\left(\mathbf{A}; W^{Conv}_{c_t,c_h,c_w}\right)\right). \tag{3}$$

In this work, we focus on modeling both long-range and short-range dependencies among video contents. Consequently, we separately elaborate on the modeling of the two groups of dependencies in the following parts.

Long-range dependency modeling. The long-range dependencies reflect the relationships of video contents viewed from a large spatial/temporal/spatio-temporal receptive field. This can be achieved by firstly setting the pooling kernel as $W^{Pool}_{T,H,W}$ for long-range spatio-temporal dependency (LST), $W^{Pool}_{T,1,1}$ for long-range temporal dependency (LT), $W^{Pool}_{1,H,W}$ for long-range spatial dependency (LS). These operations are similar to the works [19] and [46]. In this case, we can obtain three kinds of squeezed dependency features $\{\mathbf{A}^{LST} \in \mathbb{R}^{1\times1\times1\times\frac{C}{r_c}}, \mathbf{A}^{LT} \in \mathbb{R}^{1\times H\times W\times\frac{C}{r_c}}, \mathbf{A}^{LS} \in \mathbb{R}^{T\times1\times1\times\frac{C}{r_c}}\}$ by Eq. (2). Afterwards, to model the dependencies from these squeezed features, we accordingly adopt three convolutional/linear operations to mix the information across channels, yielding three corresponding dependency representations $\mathbf{R}^{LST}, \mathbf{R}^{LT}, \mathbf{R}^{LS}$ as follows

$$\begin{aligned}
\mathbf{R}^{LST} &= ReLU\left(Conv3d\left(\mathbf{A}^{LST}; W^{Conv}_{1,1,1}\right)\right), \\
\mathbf{R}^{LT} &= ReLU\left(Conv3d\left(\mathbf{A}^{LT}; W^{Conv}_{1,3,3}\right)\right), \\
\mathbf{R}^{LS} &= ReLU\left(Conv3d\left(\mathbf{A}^{LS}; W^{Conv}_{3,1,1}\right)\right),
\end{aligned} \tag{4}$$

Notablely, the linear projection is implemented by the function of $Convd3d$ with kernel $W^{Conv}_{1,1,1}$. The three dependency representations $\{\mathbf{R}^{LST}, \mathbf{R}^{LT}, \mathbf{R}^{LS}\}$ will be further reshaped to have the same size as the input feature $\mathbf{Y}'$.

Short-range dependency modeling. In contrast to the above long-range dependencies, the short-range dependency modeling shifts the focus to the information squeezed in a local spatio-temporal field. This can be achieved by setting a small receptive field for $W^{Pool}_{p_t,p_h,p_w}$. By applying the local pooling operation on the video feature map $\mathbf{Y}'$, the dynamic information presented in a local area can thus be squeezed, boosting the short-range dependency modeling. Accordingly, based on Eqs. (2) and (3), we have the local squeezed dependency feature $\mathbf{A}^S$ and excited representation $\mathbf{R}^S$. In the experiment, we empirically test three local pooling kernels, i.e., $W^{Pool}_{2,2,2}$, $W^{Pool}_{1,2,2}$ and $W^{Pool}_{1,4,4}$, yielding three squeezed features $\{\mathbf{A}^{S222}, \mathbf{A}^{S122}, \mathbf{A}^{S144}\}$ and three dependency representations $\{\mathbf{R}^{S222}, \mathbf{R}^{S122}, \mathbf{R}^{S144}\}$. We purposely use the convolution kernel $W^{Conv}_{1,1,1}$ for S222 to learn the channel interactions in $\mathbf{A}^{S222}$, by considering the temporal pooling operation with $W^{Pool}_{2,2,2}$. Differently, since there is no temporal pooling in S122 and S144 and actions in videos generally rely more on temporal modeling, we thus use a temporal convolution with the kernel $W^{Conv}_{3,1,1}$ to compute $\mathbf{R}^{S122}$ and $\mathbf{R}^{S144}\}$. We further reshape these dependency representations to the size $T \times H \times W \times \frac{C}{r_c}$ by element copying.

3.2 Dependency Aggregation

We have modeled M long-range and short-range dependencies by the MDM block. To leverage various dependencies in one residual block, the most intuitive way is averagely summing them up, as shown in Figure 3(a). The final mixed representation $\mathbf{R}_{avg} \in \mathbb{R}^{T\times H\times W\times\frac{C}{rc}}$ is as

$$\mathbf{R}_{avg} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{R}_i. \tag{5}$$

However, since different videos have different dependency preferences, simple average aggregation may neglect the important dependencies while keeping eyes on the trivial dependency. Based on this, we adopt the query structured attention (QSA) proposed in [16] to selectively combine these dependencies, which can automatically emphasize important features by larger weights. We refer this aggregation to as selective aggregation (SEC). Figure 3(b) illustrates its pipeline. Specifically, a learnable "query" vector $\boldsymbol{q} \in \mathbb{R}^{1\times\frac{C}{rc}}$ is additionally introduced to guide the attention weights calculation, and the "key" and "value" come directly from the input feature. Given the dependency representation set $\{\mathbf{R}_i\}_{i=1}^{M}$ as inputs, the attention "keys" are firstly computed by averagely pooling them across space and time, resulting in a $M \times \frac{C}{r_c}$ matrix $\mathbf{K}$. We thus can compute the attention weight of each dependency representation by

$$Attention(\boldsymbol{q}, \mathbf{K}) = softmax\left(\boldsymbol{q} \times \mathbf{K}^T\right). \tag{6}$$

And the ultimate dependency representation $\mathbf{R}_{sec} \in \mathbb{R}^{T\times H\times W\times\frac{C}{rc}}$ is thus calculated as follows

$$\mathbf{R}_{sec} = Attention(\boldsymbol{q}, \mathbf{K}) \times \mathbf{V}, \tag{7}$$

where $\mathbf{V} = [\mathbf{R}_1; \mathbf{R}_2; \cdots; \mathbf{R}_M]$. As verified in [16, 26], despite the effectiveness in such as document and video representation learning, QSA also enjoys fewer parameters than those in self-attention.

So far, we have clearly got the representation $\mathbf{R}_{sec}$ storing multiple dependencies. Next, we increase the channel number $\frac{C}{r_c}$ of $\mathbf{R}_{sec}$ to C by passing it to a 3D convolution layer with kernel $1 \times 1 \times 1$, and project the value into range (0.0, 1.0) by a Sigmoid function. Finally, we use the gating mechanism to calculate the output of SDA by

$$\mathbf{Z} = Sigmoid\left(Conv3d\left(\mathbf{R}_{sec}; 1 \times 1 \times 1\right)\right) \odot \mathbf{Y} \tag{8}$$

where $\odot$ is the Hadamard product operator.

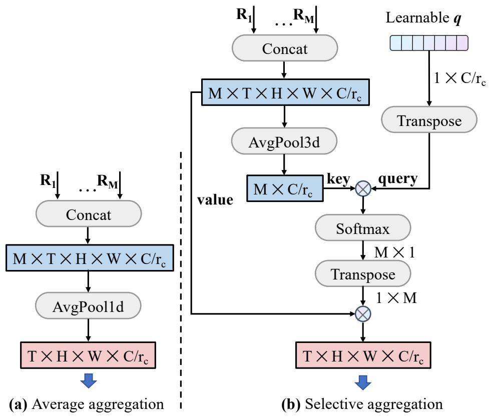

Figure 3: Diagram of dependency aggregation block.

3.3 Integrated Model and Complexity Analysis

Our proposed SDA is a plug-and-play module. We evaluate it by integrating into two simple deep video networks TSN [42] and TSM [27] that are built upon the ResNet. To illustrate the computational burden associated with the module, we first consider the comparison on number of parameters between the residual block in ResNet-50 and the SDA module. The residual block contains about $17C^2$ parameters. While, there are two main parts, i.e., the channel reduction/increase layers and the MDM block, that contain parameters in SDA[1] To be specific, the reduction/increase layers have a total of $\frac{2}{r_c}C^2$ parameters, and the sum in long-range dependency modeling part is $\frac{13}{r_c^2}$. Different pooling kernels does not lead to different number of parameters in the short-range dependency modeling part, and the value is $\frac{3}{r_c^2}$ for a single path. From the above computations, the exact number of parameters is determined by the hyperparameter r_c. In the experiment, we will specifically present the details of model complexity, including the numbers of parameter and computation burden (FLOPs).

4 EXPERIMENT

We conduct experiments on different benchmark datasets, including Something-Something V1&V2, diving-48, and EPIC-KITCHEN, for action classification. The metrics are top-1 and top-5 precision.

4.1 Datasets

Something-Something. Something-Something datasets have 174 fine-grained action categories showing humans performing predefined basic actions with everyday objects which require the ability of multiple dependencies aggregation of the model for classification. The dataset has two versions, V1 [14] and v2 [32], and contain ~110k (V1) and ~220k (V2) video clips. For the annotations of the test set is not released, we report the performance on the validation set.

Diving48. Diving48 [25] is a fine-grained video dataset of competitive diving which consists of ~18k trimmed video clips of 48 unambiguous dive sequences. As the dive sequence is composed of diverse *sub-pose*s which distribute along the timeline, the diving recognition requires multiple spatio-temporal dependencies modeling. The dataset provider has manually cleaned dive annotations with poorly segmented videos removed recently. We conduct experiments on the updated version using the latest official train/validation split V2.

Egocentric Video Datasets. EPIC-KITCHENS [8] provide researchers with kitchen actions under first-person vision, involving rich human-object interactions in daily cooking activities. We select the EPIC-KITCHENS-55 version and report the results of both verb and noun classification which have different dependency preferences, hence the classification model is required selective dependency modeling ability to perform well in the both tasks. We follow the train/validation splitting mechanism of [3]. The number of action instances in the training and validating sets are 23,191 and 5,281 respectively.

4.2 Implementation Details

We insert the SDA block into various ResNet variants like TSN [42], TSM [27]. We add a "BatchNorm" layer after each convolutional/FC layer in SDA. To avoid extremely small size of channels for MDM, we use $Max(\frac{C}{r_c}, 16)$ to limit the channel size by a minimum of 16. All models are implemented with Pytorch toolkit and run on 4× 2080Ti or 3090 GPUs.

Training. Following [42], we use uniform sampling to obtain input video frames for all datasets. For resolution, we resize the short-side of frames to 256 maintaining the aspect ratio and then crop a 224×224 patch out of resized frames. We also adopt data augmentations such as random scaling before cropping and random horizontal flipping.

We train the network with a batch size 9 per GPU. For 8-frame models, we set the learning rate (lr) as 0.01/0.015 for TSN/TSM backbone. And for 16-frame models, the lr is set to 0.015. We train 50 epochs and decay lr by 0.1 at epoch 20 and 40. The dropout ratio is set to 0.5. All backbone models are pre-trained on ImageNet.

Inference. In the ablation study, we sample one clip per video and use the center 224×224 crop for comparison. In the evaluation, we adopt testing augmentations as in [24, 27], which sample multiple clips per video, and set test resolution as 224×224 resized from 256×256 crops. We specify testing augmentations in tables.

4.3 Ablation Study

In this section, we investigate the effectiveness of various dependencies, mechanisms of dependency aggregation, and channel reduction ratio r_c by implementing ablation studies on the Something-Something V1 dataset using TSN as the backbone.

Firstly, we compare different dependency modeling mechanisms with various pooling kernels and convolutional kernels in MDM block. The two kernels jointly control the size of the receptive field, leveraging different kinds of content dependencies. Specifically, we set the pooling kernel W^{pool} as $\{T \times H \times W, T \times 1 \times 1, 1 \times H \times W\}$ for long-range dependency modeling and W^{pool} as $\{2 \times 2 \times 2, 1 \times 2 \times 2, 1 \times 4 \times 4\}$ for short-range dependency modeling. Table 1 shows the performance comparison of these kernel settings. Overall, equipping the backbone network with the content dependencies, regardless of their types, can significantly improve the performance (+8.3%-+27.2%). The results verify our claim that action categorization can be enhanced by video content dependencies. Particularly,

[1]Here, we omit the parameters in DAG block as there is only a one dimensional query vector needed to optimize.

Table 1: **Performance comparison of different kernel sizes of MDM block on the validation set of Something-Something V1. Here, we fix the hyperparameter $r_c = 4$.**

Dependency	ID	Kernel size		Acc.(%)		#P	FLOPs
		W^{Pool}	W^{Conv}	top-1	top-5		
Original TSN				19.7	46.6	23.9M	32.9G
Long-range	LST	$T \times H \times W$	$1 \times 1 \times 1$	28.0	58.2	24.6M	33.7G
	LT	$T \times 1 \times 1$	$1 \times 3 \times 3$	35.4	67.8	25.2M	33.8G
	LS	$1 \times H \times W$	$3 \times 1 \times 1$	37.1	66.2	24.7M	33.7G
	Aggregation (AVG): LST+LT+LS			46.3	74.8	25.5M	33.9G
Short-range	S222	$2 \times 2 \times 2$	$1 \times 1 \times 1$	28.2	59.0	24.6M	33.7G
	S144	$1 \times 4 \times 4$	$3 \times 1 \times 1$	44.8	74.4	24.7M	33.7G
	S122	$1 \times 2 \times 2$	$3 \times 1 \times 1$	45.9	74.8	24.7M	33.8G
	Aggregation (AVG): S222+S144+S122			45.1	74.3	25.1M	33.8G
Aggregation (AVG): LST+LT+LS+S122				46.9	75.5	25.8M	33.9G

since different actions exhibit different long-range dependencies, we observe that squeezing along space dimension (with the kernel $W^{Pool}_{1,H,W}$) and conducting temporal convolution (with the kernel $W^{Conv}_{3,1,1}$) obtains a better result than the other two long-range variants, and further averagely aggregating them achieves the higher top-1 accuracy (46.3%). Among the results of short-range dependencies, the setting of $W^{Pool}_{1,2,2}$ and $W^{Conv}_{3,1,1}$ performs the best. A small local receptive filed ($W^{Pool}_{1,2,2}$) is better than the larger one ($W^{Pool}_{1,4,4}$), and the temporal convolution ($W^{Conv}_{3,1,1}$) significantly outperform the linear projection ($W^{Conv}_{1,1,1}$). However, being different to the observation from long-range dependency modeling, the aggregation of the three short-range dependencies results in a significant performance drop, which is even worse than the single dependency counterpart (45.1% vs 45.9%). This may because that all the three variants focus on the same purpose of short-range dependency modeling. Besides the demonstrated short-range dependency modeling strategy, more complete comparison between different combinations of W^{Pool} and W^{Conv} is shown in appendix. Based on the above analysis, we also conduct a test by combing three long-range (i.e., LS, LT, LST) and one short-range dependency (S122) and get the highest performance 46.9%.

Secondly, we examine the results of different dependency aggregations, as well as the settings of hyperparameter r_c. As mentioned above, we finally select four kinds of dependencies to model in MDM. The four dependency representations will be then aggregated to form a mixup in DAG block. r_c specifies the complexity of SDA module, and here we set $r_c = 2, 4, 8, 16$. As shown in Table 2, the proposed selective aggregation method consistently outperforms the average counterpart with significant performance improvements (+0.3%-+0.6%) among all the settings of r_c. Moreover, increasing the value of r_c greatly reduces the model size (e.g., number of parameters and FLOPs) but does not degrade the performance much. Particularly, when setting $r_c = 4$, the SDA-TSN with the selective aggregation strategy achieves the performance of 47.5%, only 0.1% lower than that when $r_c = 2$. Considering the trade-off between performance and model complexity, we set $r_c = 4$ in this paper for performance report. Finally, our SDA module only introduces 8% extra parameters and 3% extra FLOPs to the original TSN model.

4.4 Comparison with State-of-the-Arts

Something-Something V1&V2. We report the top-1/top-5 performances of SDA-TSN and SDA-TSM by comparing them with the state-of-the-arts (SOTAs) in Table 3. We also list the model complexity, including the number of parameters and FLOPs, in the table.

Table 2: **Performance comparison of different dependency aggregation strategies and r_c on Something-Something V1. "AVG" denotes average aggregation and "SEC" denotes selective aggregation.**

Settings		acc.(%)		#P	FLOPs
Aggregation Stra.	r_c	top-1	top-5		
AVG	2	47.1	75.6	30.2M	35.3G
SEC		47.6	75.9	30.2M	35.4G
AVG	4	46.9	75.5	25.8M	33.9G
SEC		47.5	75.4	25.8M	33.9G
AVG	8	46.4	75.2	24.5M	33.5G
SEC		46.7	75.1	24.5M	33.5G
AVG	16	45.9	74.6	24.1M	33.3G
SEC		46.5	74.8	24.1M	33.3G

In general, our proposed SDA module enhances the performance of TSN and TSM to a new stage which is better or comparable among the competing methods. More specifically, with the inputs of 8 frames, SDA boosts the performance of TSN with a considerable absolute improvement of 27.8% (19.7%→47.5%) on V1 and 30.6% (30.0% →60.6%) on V2, making SDA-TSN competitive to the 3D spatio-temporal models such as GST, V4D, STM and SmallBig. While the improvement to TSM is relatively small: 3.0% (45.6%→48.6%) on V1 and 2.1% (59.7%→61.8%) on V2. The variance between these improvement scales lies in that TSN is built upon the standard 2D ResNet and just uses a pooling operation to achieve temporal modeling, while TSM inherently has the ability of modeling spatio-temporal relations. But from another viewpoint, it certainly offers a strong evidence that SDA successfully models multiple dependencies and greatly benefits the action classification for videos in Something-Something datasets.

Compared to the SOTAs, our SDA-TSM model attains the highest top-1 accuracy of 52.8% on V1 and 65.4% on V2 with 16 frames × 3 crops × 2 clips. The computational cost, e.g., FLOPs, is also much lower than the most advanced TEA and SmallBig methods. For example, TEA requires 70.0G×3×10=2,100G FLOPs to get the 52.3% top-1 accuracy on V1, while our SDA-TSM only needs 67.8G×3×2=406.8G FLOPs (19.4% of TEA's). In addition, as suggested in [27], we also report the ensemble results of 8-frame and 16-frame models. As shown in the table, SDA-TSN$_{En}$ and SDA-TSM$_{En}$ achieve 52.6%/66.1% and 54.8%/67.3% on Something-Something V1/V2, respectively.

Diving48. Since the new version of Diving48 has been thoroughly cleaned, we retest the performance of existing models C3D, GST, TSN and TSM for a fair comparison. Table 4 shows the performance comparison on the dataset, where all results are obtained with the input of 8 sampled frames (224×224 center crop). Compared with SOTAs that implement temporal modeling (i.e. temporal convolution or temporal shift), TSN with a simple 2D architecture also presents a relatively good result. This may suggest that the recognition of continuous diving can be achieved by a simple average combination of diving isolations along the time axis. Among SOTAs with 3d architectures, TSM performs best (77.6% top-1 accuracy) showing the good capability of action modeling. Moreover, further enhanced by SDA, TSN and TSM get remarkable absolute improvements of 7.2% and 2.6% respectively and outperforms all the SOTAs with ride margins.

Table 3: Performance comparison with state-of-the-arts on Something-Something V1 and V2 datasets.

Method	Backbone	#Pretrain	Frames×Crops×Clips	#P	FLOPs	V1		V2	
						Top-1	Top-5	Top-1	Top-5
ECO [55]	ResNet-18	Kinetics	8×1×1	47.5M	32G	39.6	—	—	—
ECO [55]			16×1×1	47.5M	64G	41.4	—	—	—
I3D [5]	3DResNet-50	ImageNet	32×1×2	28.0M	153.0G×1×2	41.6	72.2	—	—
NLI3D [43]				35.3M	168.0G×1×2	44.4	76	—	—
NLI3D+GCN [44]				62.2M	303.0G×1×2	46.1	76.8	—	—
GST [31]	ResNet-50	ImageNet	8×1×1	21.0M	29.5G×1×1	47.0	76.1	61.6	87.2
GST [31]			16×1×1	21.0M	59.0G×1×1	48.6	77.9	62.6	87.9
V4D [53]	V4DResNet-50	None	8×10×3	—	—	50.4	—	—	—
TSM+TPN [47]	ResNet-50	ImageNet	8×1×1	24.3M	33.0G×1×1	49	—	62	—
TIN [36]	ResNet-50	Kinetics	16×1×1	24.3M	67.0G×1×1	47	76.5	60.1	86.4
TEINet [28]	ResNet-50	ImageNet	8×1×1	30.4M	33.0G×1×1	47.4	—	61.3	—
TEINet [28]			16×1×1	30.4M	66.0G×1×1	49.9	—	62.1	—
RubiksNet [10]	ResNet-50	ImageNet	8×1×2	—	—	46.4	74.5	61.7	87.3
TAM [30]	ResNet-50	ImageNet	8×1×1	25.6M	33.0G×1×1	46.5	75.8	60.5	86.2
TAM [30]			16×1×1	25.6M	66.0G×1×1	47.6	77.7	62.5	87.6
TEA [24]	ResNet-50	ImageNet	8×3×10	24.5M	35.0G×3×10	51.7	80.5	—	—
TEA [24]			16×3×10	24.5M	70.0G×3×10	52.3	81.9	—	—
STM [20]	ResNet-50	ImageNet	8×3×10	24.0M	33.3G×3×10	49.2	79.3	62.3	88.8
STM [20]			16×3×10	24.0M	66.5G×3×10	50.7	80.4	64.2	89.8
SmallBig [23]	ResNet-50	ImageNet	8×3×2	—	57.0G×3×2	48.3	78.1	61.6	87.7
SmallBig [23]			16×3×2	—	114.0G×3×2	50.0	79.8	63.8	88.9
TSN [42]	ResNet-50	ImageNet	8×1×1	23.9M	32.9G	19.7	46.6	30	60.5
SDA-TSN			8×1×1	25.8M	33.9G	47.5	75.4	60.6	86.4
SDA-TSN			8×3×2	25.8M	33.9G×3 × 2	49.5	77.5	63.0	88.0
SDA-TSN			16×1×1	25.8M	67.8G	49.3	78.0	62.4	87.7
SDA-TSN			16×3×2	25.8M	67.8G×3 × 2	50.6	79.3	64.7	89.0
SDA-TSN$_{En}$			(16+8)×3×2	—	101.7G×3 × 2	52.6	80.6	66.1	89.8
TSM [27]	ResNet-50	ImageNet	8×1×1	23.9M	32.9G	45.6	74.2	59.7	86.2
SDA-TSM			8×1×1	25.8M	33.9G	48.6	77.1	61.8	87.3
TSM [27]			8×1×2	23.9M	32.9G×1×2	47.2	75.9	61.2	87.1
SDA-TSM			8×1×2	25.8M	33.9G×1×2	50.2	79.1	63.6	88.5
SDA-TSM			8×3×2	25.8M	33.9G×3×2	51.1	79.5	64.6	89.1
TSM [27]			16×1×1	23.9M	65.8G	47.2	77.1	62.0	87.6
SDA-TSM			16×1×1	25.8M	67.8G	51.3	79.6	63.3	88.5
TSM [27]			16×1×2	23.9M	65.8G×1×2	48.4	78.1	63.1	88.2
SDA-TSM			16×1×2	25.8M	67.8G×1×2	52.2	80.9	64.7	89.5
SDA-TSM			16×3×2	25.8M	67.8G×3×2	52.8	81.3	65.4	90.0
SDA-TSM$_{En}$			(16+8)×3×2	—	101.7G×3×2	**54.8**	**82.5**	**67.3**	**90.8**

Table 4: Performance comparison on the updated Diving48 dataset using the official train/validation split V2.

Method	Backbone	#Frame	Top-1	Top-5
C3D	3DResNet-50	8	73.4	96.0
GST	ResNet-50	8	74.2	94.5
TSN	ResNet-50	8	72.4	96.8
SDA-TSN	ResNet-50	8	79.6	97.4
TSM	ResNet-50	8	77.6	**97.7**
SDA-TSM	ResNet-50	8	**80.2**	97.3

Table 5: Performance comparison on EPIC-KITCHENS-55 dataset. All results are based on our train/validation split.

Method	Backbone	#Frame	Verb	Noun
C3D	3DResNet-50	8	45.2	21.5
GST	ResNet-50	8	46.4	21.1
TSN	ResNet-50	8	37.4	23.1
SDA-TSN	ResNet-50	8	**50.7**	**24.6**
TSM	ResNet-50	8	48.2	22.9
SDA-TSM	ResNet-50	8	50.0	24.4

EPIC-KITCHENS-55. To show the generality of SDA for various video recognition tasks, we also compare SDA-Nets with SOTAs on the ego-motion video dataset EPIC-KITCHENS-55 in Figure 5. The dataset focuses on cooking activities with the tasks of motion (i.e., verb) classification and object (i.e., noun) classification. We observe different changing trends of performance on the two sub-tasks. Specifically, for the verb classification, all methods with temporal modeling (i.e., C3D, GST, TSM) outperform the 2D TSN, but as for noun classification, all 3D networks fail to do better the 2D TSN model. One possible reason is that objects need more spatial modeling rather than temporal modeling. Being consistent with the observations on other datasets, SDA-Nets perform better than their backbones, which, again, demonstrates the robustness of SDA under various requirements. But, interestingly, SDA-TSN achieves a high performance of 50.7%/24.6% on the two sub-tasks, which is even slightly better than the one (50.0%/24.4%) of SDA-TSM. It could be because that there is often a severe camera shake in these epic videos and as a result, the temporal information on a small receptive field (e.g., 3) may be not precise.

4.5 Analysis and Interpretation

We investigate various dependency modeling units to understand the impact of different dependencies on different kinds of video actions using the Something-Something V1 dataset. Here, we adopt TSN as the backbone network. Figure 4 compares the per-category performances. It can be found that different types of dependencies are good at categorizing different actions. For example, as LST models the long-range spatio-temporal dependency, it can thus benefit the model for recognizing such as “Poking a stack of something so the stack collapses” that needs global dependency. LT and LS give attention to long-range temporal and spatial dependencies

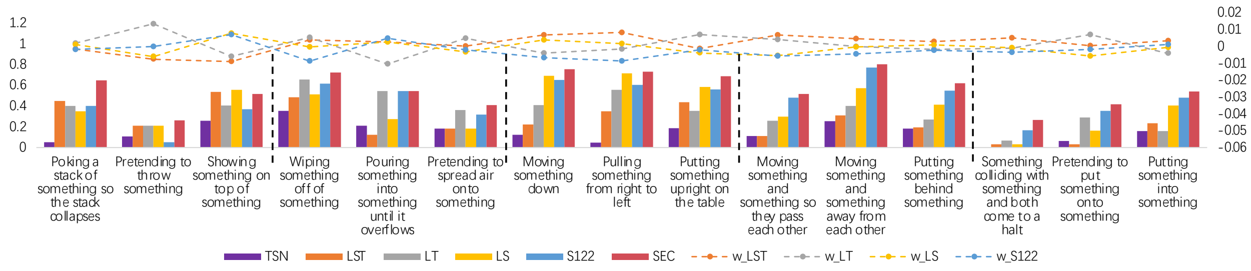

Figure 4: Per-category accuracy and centralized attention weights of different dependency modeling units over 15 action categories. The 15 actions are improved most by different SDA-TSN variants (separated by dotted lines).

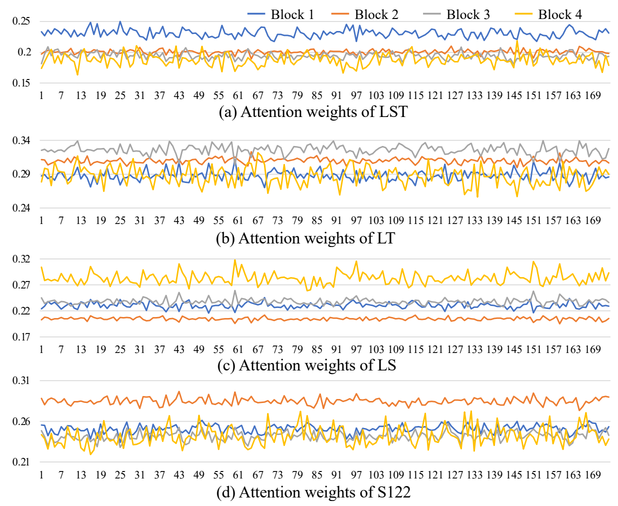

Figure 5: Average weights of dependency modeling units of SDA from Res1-4 of TSN backbone, for each of 174 categories using all validation samples in Something-Something V1. All units play a more salient role in a specific layer than others with a obvious margin of weights.

and assign the dependency impact spatially and temporally, respectively, and as a result, they can greatly boost the performance for such as "Pouring something into something until it overflows" and "Moving something down". For the actions that require strong local spatio-temporal reasoning, e.g, "Moving something and something so they pass each other", the short-range dependency modeling unit (S122) performs much better than others. Besides, the attention-based selective dependency aggregation (SEC) method consistently outperforms the single versions, demonstrating the effectiveness of dynamic multi-dependency modeling. In addition, we also show the averaged attention weights [2] of different dependencies computed by QSA, as shown by the curves in the figure. Among these actions, the importance of different dependencies are clearly distinguished, which basically proves the feasibility and utility of the used selective dependency aggregation strategy.

To more clearly understand how the dependency aggregation block (DAG) works on different residual blocks in the TSN backbone (i.e., ResNet-50), we report the averaged attention values of each dependency (e.g., LST, LT, LS, S122) computed by QSA on all the 174 action categories. Generally, a higher attention weight indicates the more important it is. From Figure 5(a), we can find that attention weights of different dependencies vary in residual blocks. Specifically, the long-range spatio-temporal dependency (LST) has a relatively larger attention value in the front layer, i.e., Residual Block 1. This phenomenon can be interpreted as that feature points in Residual Block 1 have a smaller spatio-temporal receptive field while global dependency complements the short-board. When the layer going deeper, the contribution of global dependency modeling decreases with the enlarging of the receptive field. Differently, the other two long-range dependency variants (LT and LS) impact more to the latter layers in Blocks 3 (LT) and 4 (LS) in 5(a)(b). We speculate that this is because that the low-level features learned by Blocks 1 and 2 are less precise for dependency information modeling. Moreover, the LT has a smaller attention value in Block 4. One possible explanation is that there the temporal convolutions have been stacked for at least 13 layers which results in the temporal receptive field is much larger than the tested 8 frames, decrease the importance of temporal global dependency. Similar fashion to LS but different in the pooling receptive field, the short-range modeling unit (S122) works complementarily to LS, i.e., its effect is emphasized in Block2 and inhibited in Block 4, which agrees with our expectation.

5 CONCLUSION

In this paper, we have presented a novel selective dependency aggregation module leveraging diverse dependency preferences of video contents for action classification. We firstly construct the multi-dependency modeling block to model dependencies under various perspectives by multi-direction multi-scale feature squeeze and dependency excitation. Ablation study shows that all of the dependencies can help the backbone TSN to get substantial performance improvements. Selective aggregation on those long-range and short-range dependencies further greatly boosts the performances of all backbones on various benchmarks. The visualization results of attention weights computed by the query structured attention mechanism in DAG also explicitly show the dependency preference of different video actions. Despite the substantial performance gains, SDA incurs little computation burden and few parameters to the basic networks TSN and TSM.

6 ACKNOWLEDGMENT

This research was supported by the National Key Research and Development Program of China (2020YFB1406703).

[2]To clearly demonstrate the attention weights changing trends of four dependency modeling units, we centralize them by subtracting the averages respectively.

REFERENCES

[1] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *Proceedings of the IEEE international conference on computer vision*. 5803–5812.

[2] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.

[3] Fabien Baradel, Natalia Neverova, Christian Wolf, Julien Mille, and Greg Mori. 2018. Object level visual reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 105–121.

[4] Antoni Buades, Bartomeu Coll, and J-M Morel. 2005. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 2. IEEE, 60–65.

[5] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6299–6308.

[6] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1724–1734.

[7] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

[8] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. 2018. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 720–736.

[9] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2625–2634.

[10] Linxi Fan, Shyamal Buch, Guanzhi Wang, Ryan Cao, Yuke Zhu, Juan Carlos Niebles, and Li Fei-Fei. 2020. RubiksNet: Learnable 3D-Shift for Efficient Video Action Recognition. In *European Conference on Computer Vision*. Springer, 505–521.

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

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

[13] Kunihiko Fukushima and Sei Miyake. 1982. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*. Springer, 267–285.

[14] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. 2017. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE International Conference on Computer Vision*. 5842–5850.

[15] Yanbin Hao, Zi-Niu Liu, Hao Zhang, Bin Zhu, Jingjing Chen, Yu-Gang Jiang, and Chong-Wah Ngo. 2020. Person-level Action Recognition in Complex Events via TSD-TSM Networks. In *Proceedings of the 28th ACM International Conference on Multimedia*. 4699–4702.

[16] Yanbin Hao, Hao Zhang, Chong-Wah Ngo, Qiang Liu, and Xiaojun Hu. 2020. Compact Bilinear Augmented Query Structured Attention for Sport Highlights Classification. In *Proceedings of the 28th ACM International Conference on Multimedia*. 628–636.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[19] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.

[20] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. 2019. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2000–2009.

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

[22] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1, 4 (1989), 541–551.

[23] Xianhang Li, Yali Wang, Zhipeng Zhou, and Yu Qiao. 2020. Smallbignet: Integrating core and contextual views for video classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1092–1101.

[24] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. 2020. Tea: Temporal excitation and aggregation for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 909–918.

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

[26] Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

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

[28] Zhaoyang Liu, Donghao Luo, Yabiao Wang, Limin Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Tong Lu. 2020. Teinet: Towards an efficient architecture for video recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 11669–11676.

[29] Zhenguang Liu, Kedi Lyu, Shuang Wu, Haipeng Chen, Yanbin Hao, and Shouling Ji. 2021. Aggregated Multi-GANs for Controlled 3D Human Motion Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 2225–2232.

[30] Zhaoyang Liu, Limin Wang, Wayne Wu, Chen Qian, and Tong Lu. 2020. TAM: Temporal Adaptive Module for Video Recognition. *arXiv preprint arXiv:2005.06803* (2020).

[31] Chenxu Luo and Alan L Yuille. 2019. Grouped spatial-temporal aggregation for efficient action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5512–5521.

[32] Farzaneh Mahdisoltani, Guillaume Berger, Waseem Gharbieh, David Fleet, and Roland Memisevic. 2018. On the effectiveness of task granularity for transfer learning. *arXiv preprint arXiv:1804.09235* (2018).

[33] Zhaofan Qiu, Ting Yao, and Tao Mei. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*. 5533–5541.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115, 3 (2015), 211–252.

[35] Xindi Shang, Donglin Di, Junbin Xiao, Yu Cao, Xun Yang, and Tat-Seng Chua. 2019. Annotating objects and relations in user-generated videos. In *ICMR*. 279–287.

[36] Hao Shao, Shengju Qian, and Yu Liu. 2020. Temporal interlacing network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 11966–11973.

[37] Swathikiran Sudhakaran, Sergio Escalera, and Oswald Lanz. 2020. Gate-shift networks for video action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1102–1111.

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

[39] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. 2019. Video classification with channel-separated convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5552–5561.

[40] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 6450–6459.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*.

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

[43] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2018. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7794–7803.

[44] Xiaolong Wang and Abhinav Gupta. 2018. Videos as space-time region graphs. In *Proceedings of the European conference on computer vision (ECCV)*. 399–417.

[45] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. 2018. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*. 3–19.

[46] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 305–321.

[47] Ceyuan Yang, Yinghao Xu, Jianping Shi, Bo Dai, and Bolei Zhou. 2020. Temporal pyramid network for action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 591–600.

[48] Xun Yang, Jianfeng Dong, Yixin Cao, Xun Wang, Meng Wang, and Tat-Seng Chua. [n.d.]. Tree-Augmented Cross-Modal Encoding for Complex-Query Video Retrieval. In *SIGIR, pages=1339–1348, year=2020*.

[49] Xun Yang, Fuli Feng, Wei Ji, Meng Wang, and Tat-Seng Chua. 2021. Deconfounded Video Moment Retrieval with Causal Intervention. In *SIGIR*.

[50] Xun Yang, Xueliang Liu, Meng Jian, Xinjian Gao, and Meng Wang. 2020. Weakly-supervised video object grounding by exploring spatio-temporal contexts. In *ACM MM*. 1939–1947.

[51] Xun Yang, Peicheng Zhou, and Meng Wang. 2018. Person reidentification via structural deep metric learning. *TNNLS* 30, 10 (2018), 2987–2998.

[52] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4694–4702.

[53] Shiwen Zhang, Sheng Guo, Weilin Huang, Matthew R Scott, and Limin Wang. 2019. V4D: 4D Convolutional Neural Networks for Video-level Representation Learning. In *International Conference on Learning Representations*.

[54] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. 2018. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 803–818.

[55] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. 2018. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European conference on computer vision (ECCV)*. 695–712.

A SHORT-RANGE DEPENDENCY MODELING STRATEGIES

The full comparison between different short-range dependency modeling strategies is shown in Table 6, including 3D pooling plus linear projection ($W_{1,1,1}^{Conv}$), spatial pooling plus temporal convolution, temporal pooling plus spatial convolution and pooling plus convolution along the same direction. 3D pooling plus linear projection still fall behind the strategies that adopt temporal convolution regardless of the receptive field of the pooling kernel. Spatial pooling plus temporal convolution outperforms all of other strategies and maintain the lowest computation burden and fewest parameters except 3D pooling plus linear projection, as stated in **4.3** enlarging the receptive field of spatial pooling decreases the performance. For the time length T is generally much short (e.g., 8 or 16) and even for the front layers the temporal operator in video models can easily capture the local temporal information without pooling, strategies that use temporal pooling (e.g. $Pool_{tem} + Conv_{spa}$ and $Pool_{tem} + Conv_{tem}$) boost the original TSN unobviously, comparing with their counterpart using spatial pooling. What's worse, the absence of spatial pooling leave the large spatial feature map, naturally increase the model complexity. Without any temporal feature reception, spatial pooling plus spatial convolution could not boost the original TSN as we expect.

Table 6: Performance comparison of short-range dependency modeling strategies on Something-Something V1, $r_c = 4$.

Strategy	Kernel size		Acc.(%)		#P	FLOPs
	W^{Pool}	W^{Conv}	top-1	top-5		
Original TSN			19.7	46.6	23.9M	32.9G
3D Pool	$2 \times 2 \times 2$	$1 \times 1 \times 1$	28.2	59.0	24.6M	33.7G
	$3 \times 3 \times 3$	$1 \times 1 \times 1$	29.8	61.7	24.6M	33.7G
$Pool_{spa} + Conv_{tem}$	$1 \times 2 \times 2$	$3 \times 1 \times 1$	45.9	74.8	24.7M	33.8G
	$1 \times 3 \times 3$	$3 \times 1 \times 1$	45.6	74.6	24.7M	33.8G
	$1 \times 4 \times 4$	$3 \times 1 \times 1$	44.8	74.4	24.7M	33.8G
	$1 \times 5 \times 5$	$3 \times 1 \times 1$	45.0	74.0	24.7M	33.8G
$Pool_{tem} + Conv_{spa}$	$2 \times 1 \times 1$	$1 \times 3 \times 3$	28.5	59.7	25.2M	34.2G
$Pool_{tem} + Conv_{tem}$	$2 \times 1 \times 1$	$3 \times 1 \times 1$	44.6	73.0	24.7M	33.9G
$Pool_{spa} + Conv_{spa}$	$1 \times 2 \times 2$	$1 \times 3 \times 3$	18.1	45.4	25.2M	33.9G