Dynamic Graph Modules for Modeling Object-Object Interactions in Activity Recognition

Hao Huang[1]
hhuang40@ur.rochester.edu
Luowei Zhou[2]
luozhou@umich.edu
Wei Zhang[1]
wzhang45@ur.rochester.edu
Jason J. Corso[2]
jjcorso@umich.edu
Chenliang Xu[1]
chenliang.xu@rochester.edu

[1] University of Rochester
Rochester, New York, USA
[2] University of Michigan,
Ann Arbor, Michigan, USA

Abstract

Video action recognition, a critical problem in video understanding, has been gaining increasing attention. To identify actions induced by complex object-object interactions, we need to consider not only spatial relations among objects in a single frame, but also temporal relations among different or the same objects across multiple frames. However, existing approaches that model video representations and non-local features are either incapable of explicitly modeling relations at the object-object level or unable to handle streaming videos. In this paper, we propose a novel dynamic hidden graph module to model complex object-object interactions in videos, of which two instantiations are considered: a visual graph that captures appearance/motion changes among objects and a location graph that captures relative spatiotemporal position changes among objects. Additionally, the proposed graph module allows us to process streaming videos, setting it apart from existing methods. Experimental results on benchmark datasets, Something-Something and ActivityNet, show the competitive performance of our method.

1 Introduction

Video action recognition has shown remarkable progress through the use of deep learning [4, 12, 23, 25, 26] and newly-released datasets, *e.g.*, Kinetics [13], Something-Something [9, 19], and ActivityNet [7] to name a few. Despite the importance of complex object-object interactions in defining actions (see Fig. 1 for an example), they are often overlooked. To recognize such interactions, we postulate that two general relations should be taken into consideration: 1) the interactions among different objects in a single frame, and 2) the transition of such interactions among different objects and the same object across multiple frames. We

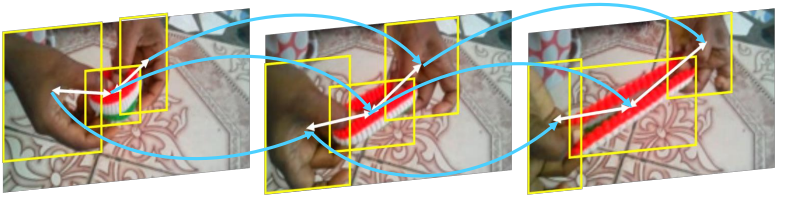


Figure 1: The action "pulling two ends of a hair band so that it gets stretched" contains interactions between two hands and a hair band. The visual graph captures the relation between visually similar objects (blue arrows) while the location graph captures relation between overlapped or close objects (white arrows).

denote the former relation as spatial relation, and the latter one as temporal relation. Both are crucial to recognize actions involving multiple objects. An effective action recognition model should be able to capture both relations precisely and simultaneously.

Despite many recent works [2, 11, 20, 27, 30] that explore modeling interactions between objects, few of them build models to capture the spatiotemporal interactions simultaneously. To model interactions among objects in both the spatial and temporal domain, we propose a dynamic graph module to capture object interactions from the beginning of a video in a progressive way to recognize actions. Similar to LSTM, we maintain a hidden state across time steps, in the form of a complete directed graph with self-connections, which we named **hidden graph**. When a new frame arrives, regions of interests (RoIs) [8, 21] in this frame are connected with nodes in the hidden graph by edges. Then, messages from RoIs in the new arriving frame will be passed to the hidden graph explicitly. After the information passing, the hidden graph further performs a self-update. A global aggregation function is applied to summarize the hidden graph for action recognition at this time step. When the next frame arrives, we repeat the above steps. Through this dynamic hidden graph structure, we capture both the spatial relation in each arrival frame and the temporal relation across frames.

To fully exploit diverse relations among different objects, we propose two instantiations of our graph module: **visual graph** and **location graph**. The visual graph is built based on the visual similarity of RoIs to link the same or similar objects and model their relations. The location graph is built on locations/coordinates of RoIs. Spatially overlapped or close objects are connected in the location graph. The **streaming** nature of our proposed methods enables the recognition of actions with only a few starting frames. As more frames come in, the accuracy of our model increases steadily. Our graph module is generic and can be combined with any 2D or 3D ConvNet in a plug-and-play fashion.

To demonstrate the effectiveness of our dynamic graph module in improving recognition performance of the backbone network, we conduct experiments on three datasets: Something-Something v1 [9], v2 [19], and ActivityNet [7]. All datasets consist of videos involving human-object interactions. Videos in Something-Something are short, trimmed, and single-labeled, while videos in ActivityNet are long, untrimmed and multi-labeled. Our experimental results support that our graph module can both process streaming videos and help improve the overall performance of existing action recognition pipelines.

2 Dynamic Graph Modules

Definition and Notations. We denote a video as $V = \{f_1, f_2, ..., f_T\}$ where f_t represents the feature map of the t-th frame extracted by a 2D ConvNet or the t-th feature map extracted by a 3D ConvNet. For each feature map, we keep its top-N region proposals generated by a Region Proposal Network (RPN) [21] and denote the set of proposals as $\boldsymbol{B}^t = \{\boldsymbol{b}_1^t, \boldsymbol{b}_2^t, \ldots, \boldsymbol{b}_N^t\}$, where the superscript denotes the frame index and the subscript indexes proposals in the current frame. We represent proposals by their coordinates and extract the associated region feature $\boldsymbol{b}_n^t \in \mathbb{R}^{1024}$ [21]. Analogous to the *hidden state* in LSTM, we maintain a *hidden graph* when chronologically processing the video, where we use proposals at $t = 1$ to initialize the hidden graph. We define the *hidden graph* as $\mathcal{G} = (\mathcal{X}, \mathcal{E})$, where $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_M\}$ denotes the set of nodes and $\mathcal{E} = \{E(\boldsymbol{x}_m, \boldsymbol{x}_k)\}$ denotes the set of weighted edges. Here, we allow self-connections within the hidden graph. Each node in the hidden graph has a feature vector and a pair of (virtual) coordinates (top-left, bottom-right). For simplicity, we also use $\boldsymbol{x_m} \in \mathbb{R}^{1024}$ to denote the feature of the m-th node in the hidden graph and use $(m_{x,1}, m_{y,1}, m_{x,2}, m_{y,2})$ to denote the coordinates of this node.

Graph Module Overview. In Fig. 2(a), we provide an unrolled version of our dynamic graph module where we omit the backbone network and RPN for simplicity. During the initialization, we use max-pooling to summarize all proposals in the first feature map as an initial context vector to warm start our graph module. For each of the following feature maps, proposals are fed into the graph module to update the structure of the hidden graph via an explicit information passing process. We design two types of hidden graphs, visual graph and location graph, based on two different dynamic updating strategies which will be elaborated in Sec. 2.1 and Sec. 2.2. At each time step, the hidden graph contains both visual features and interaction information of different regions accumulated in all previous time steps. We apply a global aggregation function to select a group of the most relevant and discriminative regions to recognize actions. More details are provided in Sec. 2.3.

2.1 The Visual Graph

Our visual graph aims to link objects with similar appearances/motions and is built based on proposal features. The graph building process is illustrated in Fig. 2(b). We use the features of top-N proposals at $t = 1$ time step to initialize the features of all nodes in the hidden graph. At time step $t > 1$, we measure the pairwise visual similarity between the N proposals in the t-th feature map and the M nodes in the hidden graph. The visual similarity is defined as:

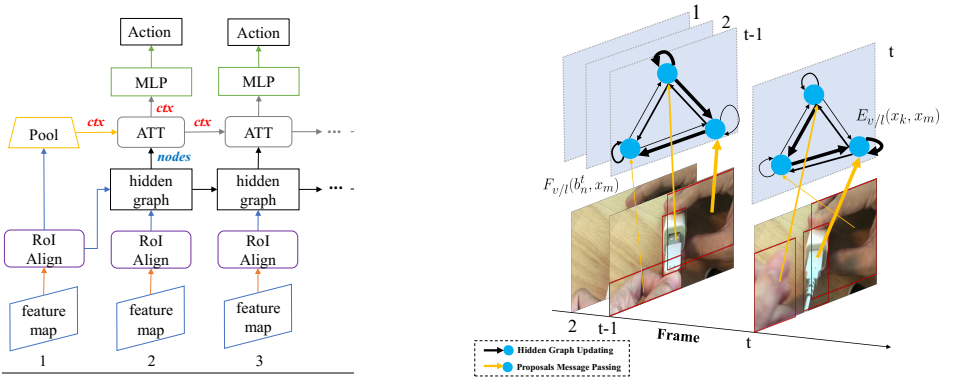
$$\boldsymbol{F_v}(\boldsymbol{b}_n^t, \boldsymbol{x}_m) = h(\boldsymbol{b}_n^t)^\top g(\boldsymbol{x}_m) \quad , \tag{1}$$

where $n = 1, 2, \ldots, N$, $m = 1, 2, \ldots, M$, and both $h(\cdot)$ and $g(\cdot)$ are linear transformations. We apply softmax$(\cdot)$ to normalize the weights of edges connecting the m-th node in the hidden graph and all proposals in the t-th feature map, so that we have:

$$\boldsymbol{F'_v}(\boldsymbol{b}_n^t, \boldsymbol{x}_m) = \frac{\exp \boldsymbol{F_v}(\boldsymbol{b}_n^t, \boldsymbol{x}_m)}{\sum_{n=1}^{N} \exp \boldsymbol{F_v}(\boldsymbol{b}_n^t, \boldsymbol{x}_m)} \quad . \tag{2}$$

Each node in the hidden graph incorporates information from all N proposals of the t-th feature map gated by $\boldsymbol{F'_v}(\boldsymbol{b}_n^t, \boldsymbol{x}_m)$. Therefore, the total amount of inflow information gathered from the t-th feature map to node m is:

$$\hat{\boldsymbol{x}}_m = \sum_{n=1}^{N} \boldsymbol{F'_v}(\boldsymbol{b}_n^t, \boldsymbol{x}_m) h(\boldsymbol{b}_n^t) \quad . \tag{3}$$



(a) The unrolled version of our graph network (backbone ConvNet and RPN are omitted). (b) The graph building process at each time step $t-1$ and t.

Figure 2: (a). A "hidden graph" is built dynamically in the temporal domain. At each time step, the hidden graph incorporates information from proposals and generates a context vector (denoted as "*ctx*" in the figure, more details in Sec. 2.3) for action recognition. (b). At $t-1$ time step, the hidden graph (top row) first incorporates messages from all proposals in the current frame (bottom row) as indicated by yellow arrows; then the hidden graph updates its edges as indicated by black arrows. The width of arrows represents the amount of information that flows along the edges. This process iterates in the following time steps.

An intuitive explanation is that each node in the hidden graph looks for the most visually similar proposals and establishes a connection based on the similarity. Subsequently, the node updates its state by absorbing the incoming information:

$$\sigma_v = \text{sigmoid}(W_m \boldsymbol{x}_m + \hat{W}_m \hat{\boldsymbol{x}}_m) \ , \quad \boldsymbol{x}_m \coloneqq \sigma_v \boldsymbol{x}_m + (1-\sigma_v)\hat{\boldsymbol{x}}_m \ , \tag{4}$$

where σ_v denotes the gate function controlled by the node state and incoming information, $W_m \in \mathbb{R}^{1024\times1024}$ and $\hat{W}_m \in \mathbb{R}^{1024\times1024}$ are learnable weights. If a proposal and a node are more visually similar in the projected space, more information will flow from this proposal to the node.

After incorporating the information from all N proposals of the t-th feature for all nodes, the hidden graph will have an internal update. Notice that the hidden graph is a complete directed graph initially including self-connections. The edge weights are computed as:

$$\boldsymbol{E}_{\boldsymbol{v}}(\boldsymbol{x}_k, \boldsymbol{x}_m) = \phi(\boldsymbol{x}_k)^\top \phi(\boldsymbol{x}_m) \ , \tag{5}$$

where $\phi(\cdot)$ is a linear layer with learnable parameters. Eq. 5 is similar to Eq. 1, except that both $\boldsymbol{x}_m$ and $\boldsymbol{x}_k$ are features of nodes in the hidden graph. After the edges of the hidden graph are updated, we propagate information for each node inside the hidden graph using a strategy similar to Eqs. 2, 3, and 4. Note that for Eqs. 2 and 3, we replace $\boldsymbol{b}_i^t$ with $\boldsymbol{x}_k$, and replace $h(\cdot)$ with $\phi(\cdot)$. Due to the different normalizations, $\boldsymbol{E}'_{\boldsymbol{v}}(\boldsymbol{x}_m, \boldsymbol{x}_k)$ differs from $\boldsymbol{E}'_{\boldsymbol{v}}(\boldsymbol{x}_k, \boldsymbol{x}_m)$, hence a directed graph. Moving to the next time step $t+1$, we repeat the above process. Taking advantage of the iterative processing, our model is capable of processing streaming videos.

2.2 The Location Graph

To utilize the displacement of objects to capture spatial relations among proposals, we propose a location graph built upon the coordinates of proposals to link objects that are over-

lapped or at close positions.

At time step t, the location-based relation between the n-th proposal in the t-th feature map and the m-th node in the hidden graph is defined as:

$$\boldsymbol{F_l}(\boldsymbol{b}_n^t, \boldsymbol{x}_m) = \boldsymbol{\sigma}_{n,m}^t \ , \tag{6}$$

where $\boldsymbol{\sigma}_{n,m}^t$ represents the value of Intersection-over-Union (IoU) between the n-th box in the t-th feature map and the m-th node in the hidden graph. Similar to [28], we adopt L-1 norm to normalize weights connecting the m-th node in the hidden graph and all proposals in the t-th feature map:

$$\boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_n^t, \boldsymbol{x}_m) = \frac{\boldsymbol{F_l}(\boldsymbol{b}_n^t, \boldsymbol{x}_m)}{\sum_{n=1}^{N} \boldsymbol{F_l}(\boldsymbol{b}_n^t, \boldsymbol{x}_m)} \ . \tag{7}$$

Analogous to the information passing process in the visual graph, each node in the hidden graph receives messages from all connected proposals from the t-th feature map:

$$\hat{\boldsymbol{x}}_m = \sum_{n=1}^{N} \boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_n^t, \boldsymbol{x}_m) p(\boldsymbol{b}_n^t) \ , \quad \boldsymbol{x}_m := \mathrm{ReLU}(\boldsymbol{x}_m + \hat{\boldsymbol{x}}_m) \ , \tag{8}$$

where $p(\cdot)$ is a linear transformation. After the information is passed from all proposals to the hidden graph, we update edges in the hidden graph dynamically. We compute IoU between each pair of nodes inside the hidden graph using Eq. 9 which is similar to Eq. 6:

$$\boldsymbol{E_l}(\boldsymbol{x}_k, \boldsymbol{x}_m) = \boldsymbol{\sigma}_{k,m} \ , \tag{9}$$

where $\boldsymbol{x}_k$ and $\boldsymbol{x}_m$ are features of nodes in the hidden graph. After the graph is built, messages can be propagated by applying Eqs. 7, 8, and 9 inside the hidden graph, where we replace $\boldsymbol{b}_i^t$ with $\boldsymbol{x}_k$, and replace $p(\cdot)$ with another linear transformation $\psi(\cdot)$.

Coordinates updating. One problem in building the location graph is how to decide the coordinates ("virtual" bounding box) of each node in the hidden graph. We propose a *coordinate shifting* strategy to approximate the coordinates of each node in the hidden graph.

We use the coordinates of the top-N proposals at time step $t = 1$ to initialize the coordinates of all nodes in the hidden graph. At time step $t > 1$, suppose the top-left and bottom-right coordinates of the m-th node in hidden graph are $(m_{x,1}^{t-1}, m_{y,1}^{t-1}, m_{x,2}^{t-1}, m_{y,2}^{t-1})$, and the coordinates of the n-th proposal in the t-th feature map are $(n_{x,1}^t, n_{y,1}^t, n_{x,2}^t, n_{y,2}^t)$. The normalized weight (IoU) between the m-th node in the hidden graph and the n-th proposal in the t-th feature map is $\boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_n^t, \boldsymbol{x}_m)$. The larger the weight, the more information will flow from the n-th proposal to the m-th node and the coordinates of the m-th node will shift more towards the position of the n-th proposal. After information passing, the target position of the m-th node is the center of the current position and the weighted average positions of all proposals in the t-th feature map connected to the m-th node. Formally, the coordinate of $m_{x,1}^t$ is computed as:

$$m_{x,1}^t = \frac{1}{2}(m_{x,1}^{t-1} + \sum_{n=1}^{N} \boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_n^t, \boldsymbol{x}_m) n_{x,1}^t) \ , \tag{10}$$

Similarly for $m_{y,1}^t$, $m_{x,2}^t$ and $m_{y,2}^t$ which can be found in the Appendix. Hence, coordinates attached to nodes in the hidden graph will update dynamically according to input proposals at each time step.

2.3 Attention on Graph

At each time step, the hidden graph contains accumulated information from all preceding time steps. The recognition decision is generated based on the state of the hidden graph. We need an aggregation function ρ to gather information from all nodes in the hidden graph. At the same time, such a function should be invariant to permutations of all nodes [3].

Attention mechanism was first proposed in [1] and it takes a weighted average of all candidates based on a query [14]. We add a virtual node to summarize the hidden graph at each time step (see the "ATT" block in Fig. 2(a)). The feature of this virtual node serves two purposes: one is to recognize actions at the current time step, and another is to act as a query (or context) to aggregate information from the hidden graph at the next time step. Specifically, the feature of virtual node at time step t is denoted as $\boldsymbol{q}_t$, the feature of m-th node in hidden graph at time step $t+1$ is denoted as $\boldsymbol{x}_m^{t+1}$, then the feature of virtual node at time step $t+1$, denoted as $\boldsymbol{q}_{t+1}$, is computed as:

$$e_m^{t+1} = \tanh\left(W_g \boldsymbol{q}_t + W_h \boldsymbol{x}_m^{t+1}\right), \quad \alpha_m^{t+1} = \frac{\exp W_o e_m^{t+1}}{\sum_{m=1}^{M} \exp W_o e_m^{t+1}}, \quad \boldsymbol{q}_{t+1} = \sum_{m=1}^{M} \alpha_m^{t+1} \boldsymbol{x}_m^{t+1}, \quad (11)$$

where W_c, W_h and W_o are learnable weights. Note that the initial feature of the virtual node is the max-pooling of all proposals in the first feature map. Once the feature of the virtual node is generated, we can forward the feature into a multi-layer perceptron to recognition actions.

3 Full Model for Action Recognition

In this section, we introduce two versions of our full models: streaming version and static version. The streaming version can process streaming videos while the static version incorporates the global video feature and achieves better overall performance.

Streaming Version. Given a video clip (around 5 seconds), our model first randomly samples 32 frames. The sampled frames are fed into a backbone network. In our case, we apply a 3D ConvNet [4]. The output of the backbone is a sequence of 3D feature maps with the shape of $T \times C \times H \times W$. We apply a region proposal network (RPN) [21] to extract proposals for each sampled frame. With the proposed bounding boxes, we conduct RoIAlign [10] on the sequence of feature maps. We build our graph module dynamically upon a sequence of RoI proposals from the feature maps. We maintain a "hidden graph" which evolves along the temporal dimension and generates a recognition result at each time step.

Static Version. To achieve better recognition accuracy, it is beneficial to utilize all information contained in a video. We provide a static version of our model in which we sample frames from an entire video and input all sampled frames into both the backbone 3D ConvNet and a RPN. We average pool the features produced by the 3D ConvNet from $T \times C \times H \times W$ to $C \times 1$, denoted as $\boldsymbol{f}$. Different from the streaming version where we only use the graph module feature $\boldsymbol{q}_t$ at each time step, here we fuse both graph module features and 3D ConvNet features by concatenating them to recognize actions. More details about the fusion layers are in the Appendix.

4 Experiment

4.1 Datasets, Metrics, and Implementation Details

Datasets. We evaluate our dynamic graph module on datasets: Something-Something v1 and v2 [9, 19] and ActivityNet [7]. Something-Something v1 [9] contains more than 100K short videos and v2 [19] contains around 220K videos. The average video duration is about 3 to 6 seconds. There are 174 total action classes and each video corresponds to exactly one action. For both v1 and v2 datasets, we follow the official split to train and test our model. ActivityNet contains 10K videos for training, enclosing 15K activity instances from 200 activity classes. The validation set contains 5K videos and 7K activity instances. We also follow the official split to train and test our model.

Metrics. Since all videos in Something-Something dataset are single-labeled, we adopt recognition accuracy (top-k) as our evaluation metrics. In ActivityNet dataset, mean average precision (mAP) is also used for prediction evaluation as some videos have multiple labels.

Compared methods. To verify that our dynamic graph module is capable of modeling interactions between objects, we design a baseline LSTM model where we feed in the mean-pooled top-N region features at each time step. We compare our streaming model with this baseline, along with a state-of-the-art method [33]. We also compare our full static model with competitive existing works [16, 17, 22, 28, 32, 33].

Region Proposal and Feature. For each input frame, we propose RoI proposals using RPN with ResNet-50 pre-trained on Microsoft COCO. We project proposal coordinates from the input frames back to the feature maps generated by the penultimate convolutional block of 3D backbone. Since 32 input frames are reduced to 8 feature maps in the temporal domain, we select 8 input frames (i.e., 1-th, 5-th, 9-th, ...) to match the 8 feature maps. We apply RoIAlign [10] with the same configuration in [28] to extract features for each proposal.

Training. For the backbone network, we follow the frame sampling and training strategy in [28]. Then for our full model, we fix the backbone 3D ConvNet and only train other parts, *e.g.*, our graph module, fusion layers and classification layer. We adopt the same learning strategy as the fine-tuning of the backbone. More details are in the Appendix.

Inference. For Something-Something dataset, we uniformly sample 32 frames from the entire video and rescale them with the shorter side to 256. Then we center crop each frame to 224×224. For ActivityNet dataset, we segment each video into 5s long clips without overlapping and uniformly sample 32 frames from each clip. We adopt top-k pooling to average scores of all clips as the video-level score.

4.2 Results of Streaming Model

Videos in the Something-Something dataset usually contain two to three objects including humans. We keep the top 20 region proposals for each frame and fix the number of nodes in the hidden graph to 5. We plot the top-1 accuracy in Fig. 3.

The accuracy of the baseline model is significantly lower than any of our graph modules, indicating that feeding the average pooling over proposals into an LSTM fails to capture interactions between objects. One possible explanation is that the average pooling operation discards the spatial relations contained in proposals. The only temporal relation modeled by LSTM is insufficient to capture interactions. On the contrary, as our graph module maintains a graphical structure to keep both spatial and temporal relations among proposals, it has the capability to model the complex interactions among objects.

Model	Someth. v1		Someth. v2	
Ours	Visual	Location	Visual	Location
	41.7	38.2	54.0	50.5
ECO Lite [33]	41.3		-	

Table 1: Top-1 accuracy of the last feature map on Someth. validation set. ("Visual" and "Location" refer to "visual graph" and "location graph".)

model	mAP	top-1	top-3
Backbone	70.2	69.2	83.5
Visual Graph	71.5	**70.3**	84.5
Location Graph	**71.8**	**70.3**	84.8

Table 2: Results of the static version model on ActivityNet dataset. With RGB frame inputs only.

Between the two graph modules, we notice that the visual graph outperforms the location graph. That is possibly because the visual graph contains more parameters than the location graph, which gives the visual graph more powerful modeling ability. Though the location graph performs inferior than the visual graph, it still achieves more than 38.5% top-1 accuracy. It is reasonable to conclude that the graph module structure intrinsically has the ability to model interactions regardless of any specific instantiation.

The accuracy of the two graph modules increases steadily as the number of frames increases and plateaus at 7-th feature map. It demonstrates that our graph module has the ability to recognize actions in streaming videos, even if only parts of frames are forwarded into the module. We also report the accuracy of the last feature map in Table 1. On Something-Something v1 dataset, our visual graph performs slightly better than [33] which is a recent state-of-the-art streaming method. Distinct from [33], our model explicitly focuses on modeling object interactions. The location graph does not perform as competitive as the visual graph. However, note that the location graph has fewer parameters as illustrated in Sec. 2.2.

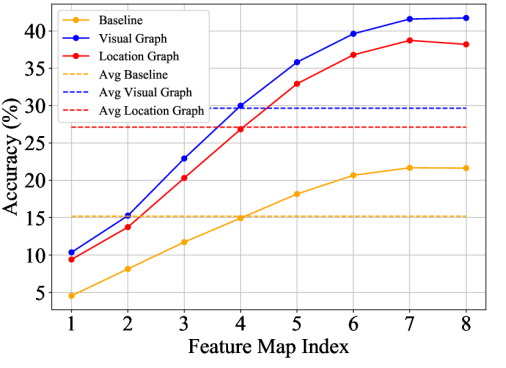


Figure 3: Top-1 accuracy on Something-Something v1 validation set for each feature map. "Avg" means the average accuracy of the total eight feature maps.

4.3 Results of Static Model

Something-Something Dataset. We compare our static version model with some recent works [16, 17, 22, 28, 32, 33] shown in Table 3. For Something-Something v1 validation set, the backbone 3D ConvNet has achieved 46.0% in top-1 accuracy and 76.1% in top-5 accuracy. By adding our two types of dynamic graph modules to the backbone, the performance improves an absolute 1.1%. For Something-Something v2 validation set, the backbone 3D ConvNet has achieved 59.7% in top-1 and 86.4% in top-5 accuracy. Our graph module still boosts the performance of the backbone by an absolute 1.7% for top-1 accuracy. We also report our results on the leaderboard (results shown in the "test" column). Without bells and whistles (*e.g.*, flow inputs and ensembling), our model achieves competitive results.

ActivityNet Dataset. We also evaluate our static version model on ActivityNet dataset and report the result in Table 2. Different from trimmed and shorter videos in Something-Something dataset, videos in ActivityNet are untrimmed and longer, and some contain mul-

		Something v1			Something v2			
		val		test	val		test	
	Modality	top-1	top-5	top-1	top-1	top-5	top-1	top-5
2-Stream TRN [32]	Flow+RGB	42.0	-	40.7	55.5	83.1	56.2	83.2
MFNet-C101 [16]	RGB only	43.9	73.1	37.5	-	-	-	-
Space-Time Graphs [28]	RGB only	46.1	**76.8**	**45.0**	-	-	-	-
$ECO_{En}Lite$ [33]	RGB only	46.4	-	42.3	-	-	-	-
$ECO_{En}Lite$ [33]	Flow+RGB	**49.5**	-	43.9	-	-	-	-
TSM_{16F} [17]	RGB only	44.8	74.5	-	58.7	84.8	**59.9**	**85.9**
LEGO [22]	RGB only	45.9	-	-	59.6	-	-	-
Backbone	RGB only	46.0	76.1	-	59.7	86.4	-	-
Visual Graph	RGB only	**47.1**	76.2	-	**61.4**	**86.8**	-	-
Location Graph	RGB only	**47.1**	**76.3**	**44.5**	**61.4**	**86.8**	**59.7**	**86.1**

Table 3: Comparing performance of the static version model on Something-Something v1 & v2 datasets with state-of-the-art methods. The "test" columns are leaderboard results. Note that we only use RGB modality and relatively simple preprocessing steps. The top two scores of each metric are highlighted. ("-" means there is no publicly available evaluation scores released by the authors.)

tiple actions. The backbone 3D ConvNet has achieved 69.2% top-1 accuracy and the mAP is 70.2%[1]. Note that compared with the state-of-the-art performance [26, 31], we only apply random rescale and random horizontal flip to RGB images without any other complicated data augmentation. We also do not use audio modality, optical-flow features or ensembles, etc. Both types of dynamic graph modules bring around 1.5% improvement in mAP compared to the backbone. The result demonstrates our module's capability on long-term action recognition in untrimmed videos. As our model is trained on trimmed action instances level by sampling a fixed number of frames but tested on whole videos, we can draw a conclusion that our proposed graph module is capable of recognizing actions in both single-labeled trimmed videos and multi-labeled untrimmed videos.

5 Related Works

Video action recognition with deep learning. Many works have applied convolutional networks to tackle video action recognition problems [4, 12, 23, 24, 25, 26, 29]. Karpathy *et al*. [12] explored various approaches of fusing RGB frames in temporal domain. Simonyan *et al*. [23] devised a two-stream model to fuse RGB features and optical flow features. Tran *et al*. [24] applied a 3D kernel to convolve a sequence of frames in spatiotemporal domain. [4] proposed inflated 3D convolutional networks (I3D) which utilize parameters in 2D ConvNets pre-trained on ImageNet [15]. [26] proposed the temporal segment network (TSN) which sparsely sampled frames. Zhou *et al*. [32] showed that the order of frames is crucial for correct recognition. Zolfaghari *et al*. [33] proposed an online video understanding system combining 2D ConvNets and 3D ConvNets. ConvNets is also one of the components in our model. However, ConvNets lack the power to model explicit object interactions, which is the problem the proposed module aims to solve.

Relational model / Graph neural networks. Another line of work in action recognition is focusing on modeling object relationships. Ma *et al*. [18] utilized an LSTM to model object interactions but lost spatial information. Wang *et al*. [29] added a non-local layer

[1]The 3D backbone network is our own implementation.

to 3D ConvNets to capture relations among different positions in feature maps. However, two distant positions are generally likely to be irrelevant. Some works apply graph neural network (GNN) to model object relations. [5] projected pixels to graph space and then projected back to build relations among different regions, but it cannot guarantee each region corresponds to (a part of) an object. The most similar work to ours is [28], where a video is represented as a global space-time graph of object regions. We propose to use a dynamic hidden graph to process sequential video input, in the form of object region proposals, which takes advantage of both relational modeling and sequential modeling [6].

6 Conclusion

We propose a novel dynamic graph module with two instantiations, visual graph and location graph, to model object-object interactions in video activities. By considering object relations in spatial and temporal domains simultaneously, the proposed graph module can capture interactions among objects explicitly in streaming video settings, which differs our work from existing methods. We will extend our graph module to more sequential modeling fields, *e.g.* video prediction, in the future.

Acknowledgement. H. Huang, W. Zhang, and C. Xu are supported by NSF IIS 1813709, IIS 1741472, and CHE 1764415. L. Zhou and J. J. Corso are supported by DARPA FA8750-17-2-0125, NSF IIS 1522904 and NIST 60NANB17D191. This article solely reflects the opinions and conclusions of its authors but not the DARPA, NSF, or NIST.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4724–4733. IEEE, 2017.

[5] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Shuicheng Yan, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. *arXiv preprint arXiv:1811.12814*, 2018.

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

[7] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.

[8] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[9] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. TheâĂİ something somethingâĂİ video database for learning and evaluating visual common sense. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 1, page 3, 2017.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[11] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018.

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

[13] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[14] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Myunggi Lee, Seungeui Lee, Sungjoon Son, Gyutae Park, and Nojun Kwak. Motion feature network: Fixed motion filter for action recognition. In *European Conference on Computer Vision*, pages 392–408. Springer, 2018.

[17] Ji Lin, Chuang Gan, and Song Han. Temporal shift module for efficient video understanding. *arXiv preprint arXiv:1811.08383*, 2018.

[18] Chih-Yao Ma, Asim Kadav, Iain Melvin, Zsolt Kira, Ghassan AlRegib, and Hans Peter Graf. Attend and interact: Higher-order object interactions for video understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6790–6800, 2018.

[19] Farzaneh Mahdisoltani, Guillaume Berger, Waseem Gharbieh, David Fleet, and Roland Memisevic. Fine-grained video classification and captioning. *arXiv preprint arXiv:1804.09235*, 2018.

[20] Siyuan Qi, Wenguan Wang, Baoxiong Jia, Jianbing Shen, and Song-Chun Zhu. Learning human-object interactions by graph parsing neural networks. In *European Conference on Computer Vision*, pages 407–423. Springer, 2018.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[22] Jie Shao, Kai Hu, Yixin Bao, Yining Lin, and Xiangyang Xue. High order neural networks for video classification. *arXiv preprint arXiv:1811.07519*, 2018.

[23] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[24] Du Tran, Lubomir D Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3d: generic features for video analysis. *CoRR, abs/1412.0767*, 2(7):8, 2014.

[25] Du Tran, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri. Convnet architecture search for spatiotemporal feature learning. *arXiv preprint arXiv:1708.05038*, 2017.

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

[27] Limin Wang, Wei Li, Wen Li, and Luc Van Gool. Appearance-and-relation networks for video classification. *arXiv preprint arXiv:1711.09125*, 2017.

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

[29] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[30] Nicholas Watters, Andrea Tacchetti, Theophane Weber, Razvan Pascanu, Peter Battaglia, and Daniel Zoran. Visual interaction networks. *arXiv preprint arXiv:1706.01433*, 2017.

[31] Yuanjun Xiong, Limin Wang, Zhe Wang, Bowen Zhang, Hang Song, Wei Li, Dahua Lin, Yu Qiao, Luc Van Gool, and Xiaoou Tang. Cuhk & ethz & siat submission to activitynet challenge 2016. *CoRR*, abs/1608.00797, 2016.

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

[33] Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 695–712, 2018.

A Appendix

This Appendix provides additional algorithm formulas, network structures and implementation details.

A.1 Coordinates updating

At time step $t > 1$, suppose the top-left and bottom-right coordinates of the m-th node in hidden graph are $(m_{x,1}^{t-1}, m_{y,1}^{t-1}, m_{x,2}^{t-1}, m_{y,2}^{t-1})$, and the coordinates of the n-th proposal in the t-th feature map are $(n_{x,1}^{t}, n_{y,1}^{t}, n_{x,2}^{t}, n_{y,2}^{t})$. The normalized weight (IoU) between the m-th node in the hidden graph and the n-th proposal in the t-th feature map is $\boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_n^t, \boldsymbol{x}_m)$. The coordinate of $m_{x,1}^t$ is computed as:

$$\begin{cases} m_{x,1}^t = \frac{1}{2}(m_{x,1}^{t-1} + \sum_{n=1}^{N} \boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_n^t, \boldsymbol{x}_m) n_{x,1}^t) \ , \\ m_{y,1}^t = \frac{1}{2}(m_{y,1}^{t-1} + \sum_{n=1}^{N} \boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_n^t, \boldsymbol{x}_m) n_{y,1}^t) \ , \\ m_{x,2}^t = \frac{1}{2}(m_{x,2}^{t-1} + \sum_{n=1}^{N} \boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_n^t, \boldsymbol{x}_m) n_{x,2}^t) \ , \\ m_{y,2}^t = \frac{1}{2}(m_{y,2}^{t-1} + \sum_{n=1}^{N} \boldsymbol{F}_{\boldsymbol{l}}'(\boldsymbol{b}_i^t, \boldsymbol{x}_m) n_{y,2}^t) \ . \end{cases} \tag{12}$$

A.2 The Structure of Fusion Layers

The average-pooled feature produced by the 3D ConvNet is denoted as $\boldsymbol{f} \in \mathbb{R}^{C\times 1}$ where $C = 2048$. The graph module feature $\boldsymbol{q}_t \in \mathbb{R}^{C'\times 1}$ where $C' = 1024$. We fuse both graph module feature and 3D ConvNet feature to recognize actions. The fusion layers are illustrated in Fig. 4. We keep the size of the fused feature $\boldsymbol{z}_t$ to $C' \times 1$ and forward this feature into a multi-layer perceptron to get the final recognition results.

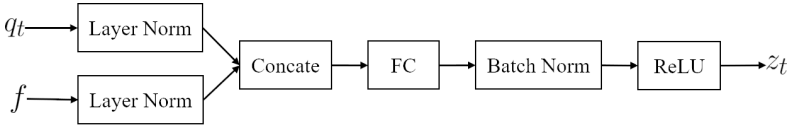


Figure 4: Fusion layers to fuse the graph module feature and 3D ConvNet feature at time step t.

A.3 Implementation Details

We first train our backbone 3D model [4, 29] on Kinetics dataset and then fine-tune it on the target datasets. For Something-Something dataset, we randomly sample 32 frames from each video. For ActivityNet dataset, as the video length is much longer, we first segment each activity instance into several clips (around 5 seconds) with the overlap rate fixed to 20%. The sampled frames are used to train our backbone 3D model. Following [28], sampled frames are randomly scaled with shorter side resized to a random integer number in [256, 320]. Then we randomly crop out an area of 224×224 and randomly flip frames horizontally before forwarding them to the backbone model. The Dropout [**?**] before the classification layer in backbone model is set to 0.5. We train our backbone model with a batch size of 24. We set the initial learning rate to 0.00125. We apply stochastic gradient descent (SGD)

optimizer and set momentum to 0.9 and weight decay to 0.0001. We adopt cross-entropy loss during our training. We adopt cross-entropy loss during our training.

Next, we describe how we train our streaming dynamic graph module. For each input frame, we propose RoI proposals using RPN [21] with ResNet-50 pre-trained on Microsoft COCO. For Something-Something dataset, we keep the top 20 proposals each frame and set the number of nodes in hidden graph to be 5. For ActivityNet dataset, as video scenes are more complex and contain more objects, we keep the top 40 proposals and increase the number of graph nodes to 10. We fix the backbone 3D ConvNet and only train our graph module, fusion layers and classification layer. We adopt the same learning strategy as the fine-tuning of the backbone.

For the static model, we first train the streaming model following the strategy above for 3 epochs as a warm-up. Then we concatenate the graph module feature with the backbone feature using the fusion layers described in Sec. A.2. At the same time, we reduce the learning rate by a factor of 10. The parameters of the backbone remain fixed during training.