VideoGraph: Recognizing Minutes-Long Human Activities in Videos

Noureldien Hussein, Efstratios Gavves, Arnold W.M. Smeulders
QUVA Lab, University of Amsterdam
{nhussein, egavves, a.w.m.smeulders}@uva.nl

Abstract

Many human activities take minutes to unfold. To represent them, related works opt for statistical pooling, which neglects the temporal structure. Others opt for convolutional methods, as CNN and Non-Local. While successful in learning temporal concepts, they fall short of modeling minutes-long temporal dependencies. We propose VideoGraph, a method to achieve the best of two worlds: represent minutes-long human activities and learn their underlying temporal structure. To represent human activities, VideoGraph learns a soft version of an undirected graph. The graph nodes are deterministic and are learned entirely from video datasets, making VideoGraph applicable to video understanding tasks without node-level annotation. While the graph edges are probabilistic and are learned in a soft-assignment manner. The result is improvements over related works on benchmarks: Breakfast, Epic-Kitchens and Charades. Besides, we demonstrate that VideoGraph is able to learn the temporal structure of human activities in minutes-long videos.

1. Introduction

Human activities in videos can take many minutes to unfold, each is packed with plentiful of fine-grained visual details. Take for example two activities: "making pancake" or "preparing scrambled eggs". The question is what makes a difference between these two activities? Is it the fine-grained details in each, or the overall painted picture by each? Or both?

The goal of this paper is to recognize minutes-long human activities as defined by [1], also referred to as complex actions in [2]. A long-range activity consists of a set of unit-actions [1], also known as one-actions [2]. For example, the activity of "making pancakes" includes unit-actions: "cracking egg", "pour milk" and "fry pancake". Some of these unit-actions are crucial to distinguish the activity. For example, the unit-action "cracking egg" is all what needed to discriminate the activity of "making pancakes" from "preparing coffee". Also, long-range activity is recognized only in its entirety, as its unit-actions are insufficient by themselves. For example, only a short video snippet of unit-action "cracking egg" cannot tell apart "making pancake" from "preparing scrambled eggs", as both activities share the same unit-action "cracking egg". Added to this, the temporal order of unit-actions for a specific activity may be permuted. There exist different orders of how we can carry out an activity, like "prepare coffee", see figure 1. Nonetheless, there exist some sort of temporal structure for such activity. One can start "preparing coffee" by "taking cup" and usually end up with "pour sugar" and "stir coffee". So, to recognize long-range human activities, goals to be met are: modeling the temporal structure of the activity in its entirety, and occasionally paying attention to its fine-grained details.

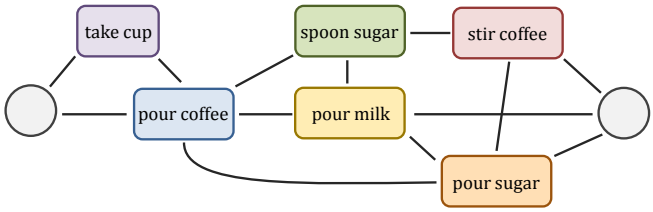


Figure 1: The activity of "preparing coffee" can be represented as undirected graph of unit-actions. We are inspired by graphs to represent this activity. The reason is that a graph can portray the many ways one can carry out such activity. More over, it preserves the temporal structure of the unit-actions. Reproduced from [1].

There exist two distinct approaches for long-range temporal modelling. The first approach is orderless modeling. Statistical pooling [3] and vector encoding [4, 5] are used to aggregate video information over time. The upside is the ability to address seemingly minutes- or even hours-long videos. The downside, however, is the inability to learn temporal patterns and the arrow-of-time [6]. Both are proven to be crucial for some tasks [7, 8]. The second approach is order-ware modelling. 3D CNN is proven to be successful in learning spatiotemporal concepts for short video snippets with strict temporal pattern [9]. Careful design choices enable them to model up to minute-long tem-

poral dependencies [2]. But for minutes-long human activities, the strict temporal pattern no longer exists. So, the question arises: how to model the temporal structure of minutes or even hour-long human activities?

This paper proposes VideoGraph, a graph-inspired representation to achieve the aforementioned goal. A soft version of undirected graph in learned completely from the dataset. The graph nodes represent the key latent concepts of which the human activity is composed. These latent concepts are analogous to one-actions. While the graph edges represent the temporal relationship between these latent concepts, *i.e.* the graph nodes. VideoGraph has the following novelties. *i.* In its graph-inspired representation, VideoGraph models human activity for up to thirty-minute videos, whereas the state-of-the-art is one minute [2]. *ii.* A proposed node embedding block to learn the graph nodes from data. This circumvents the node annotation burden for long-range videos, and makes VideoGraph extensible to video datasets without node-level annotation. *iii.* A novel graph embedding layer to learn the relationships between graph nodes. The outcome is representing the temporal structure of long-range human activities. The result is achieving improvements on benchmarks for human activities: Breakfast [1], Epic-Kitchens [10] and Charades [11].

2. Related Work

Orderless *v.s.* Order-aware Temporal Modeling. Be it short-, mid-, or long-range human activities, when it comes to temporal modeling, related methods are divided into two main families: orderless and order-aware. In orderless methods, the main focus is the statistical pooling of temporal signals in videos, without considering their temporal order or structure. Different pooling strategies are used, as max and average pooling [3], attention pooling [12], and context gating [13], to name a few. A similar approach is vector aggregation, for example: Fisher Vectors [14] and VLAD [4, 5]. Although statistical pooling can trivially scale up to extremely long sequences in theory, this comes at a cost of losing the temporal structure, reminiscent of Bag-of-Words losing spatial understanding.

In order-aware methods, the main attention is payed to learning structured or ordered temporal patterns in videos. For example, LSTMs [15, 16], CRF [17], 3D CNNs [18, 9, 19, 20, 21]. Others propose temporal modeling layers on top of backbone CNNs, as in Temporal-Segments [22], Temporal-Relations [23] and Rank-Pool [24]. The outcome of order-aware methods is substantial improvements over their orderless counterparts in standard benchmarks [25, 26, 27]. Nevertheless, both temporal footprint and computational cost remain the main bottlenecks to learn long-range temporal dependencies. The best methods [2, 21] can model as much as 1k frames (∼30 seconds), which is a no match to minutes-long videos. This paper strives for the best of two worlds: learning the temporal structure of human activities in minutes-long videos.

Short-range Actions *v.s.* Long-range Activities.

Huge body of work is dedicated to recognizing human actions that take few seconds to unfold. Examples of well-established benchmarks are: Kinetics [25], Sports-1M [28], YouTube-8M [29], Moments in Time [30], 20B-Something [31] and AVA [32]. For these short- or mid-range actions, [7] demonstrates that a few frames suffice for a successful recognition. Other strands of work shift their attention to human activities that take minutes or even an hour to unfold. Cooking-related activities are good examples, as in YouCook [33], Breakfast [1], Epic-Kitchens [10], MPII Cooking [34] or 50-Salads [35]. Other examples include instructional videos: Charades [11], and unscripted activities: EventNet [36], Multi-THUMOS [37].

In all cases, several works [1, 2, 34, 38] define the differences between short- and long-range human actions, albeit with a different naming or terms. We follow the same definition of [1]. More formally, we use *unit-actions* to refer to fine-grained, short-range human actions, and *activities* to refer to long-range complex human activities.

Graph-based Representation. Earlier, graph-based representation has been used in storytelling [39, 40], and video retrieval [41]. Different works use graph convolutions to learn concepts and/or relationships from data [42, 43, 44]. Recently, graph convolutions are applied to image understanding [45], video understanding [46, 47, 48, 49] and question answering [50]. Despite their success in learning structured representations from video datasets, the main limitation of graph convolution methods is requiring the graph nodes and/or edges to be known a priori. Consequently, when node or frame-level annotations are not available, using these methods is hard. In contrast, this paper aims for a graph-inspired representation in which the graph nodes are fully inferred from data. The result is that our paper is extensible to datasets without node-level annotations.

Self-Attention is used extensively in language understanding [51]. The recently proposed the transformer block shows substantial improvements in machine translation [52], image recognition [21] and video understanding [48, 53] or even graph representations [54]. The transformer block [53] attends to a local feature conditioned on both local and global context. That is why it outperforms the self-attention mechanism [55, 56, 57], which is conditioned on only the local feature.

A video of human activity consists of short snippets of unit-actions. This paper is inspired by all these attention mechanisms to attend to a unit-action (*i.e.* local feature) based on the surrounding activity (*i.e.* global context).

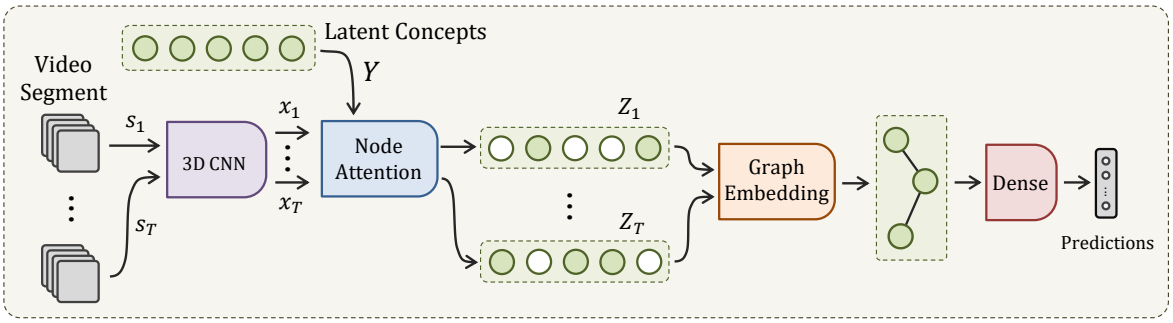


Figure 2: Overview of VideoGraph. It takes as input a video segment s_i of 8 frames from an activity video v. Then, it represents it using standard 3D CNN, *.e.g* I3D. The corresponding feature representation is x_i. Then, a node attention block attends to a set of N latent concepts based on their similarities with x_i, which results in the node-attenative representation Z_i. A novel graph embedding layer then processes Z_i to learn the relationships between its latent concepts, and arrives at the final video-level representation. Finally, an MLP is used for classification.

3. Method

Motivation. We observe that a minutes-long and complex human activity usually is sub-divided into unit-actions. Similar understanding is concluded by [1, 2], see Fig. 1. So, one can learn the temporal dependencies between these unit-actions using methods for sequence modeling in videos, as LSTM [15] or 3D CNN [20]. However, these methods face the following limitations. First, such activities may take several minutes or even hours to unfold. Second, as video instances of the same activity are usually wildly different, there is no single temporal sequence that these methods can learn. For example, one can "prepare coffee" in many different ways, as the various paths in Fig. 1 indicate. Nevertheless, there seems to be an over-arching weak temporal structure of unit-actions when making a coffee.

We are inspired by graphs to represent the temporal structure of the human activities in videos. The upside is the ability of a graph-based representation to span minutes- or even hour-long temporal sequence of unit-actions while preserving their temporal relationships. The proposed method, VideoGraph, is depicted in Fig. 2, and in the following, we discuss its details.

VideoGraph.

We start from a video v comprising T randomly sampled video segments $v = \{s_i \,|\, i = 1, 2, ..., T\}$. Each segment s_i is a burst of 8 successive video frames, and represented as feature $x_i \in \mathbb{R}^{1\times H\times W\times C}$ using standard 3D CNN, for example I3D [9], where C is the number of channels, H, W are height and width of the channels. Our goal is to construct an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ to represent the structure of human activity in video v. The graph nodes $\mathcal{N}$ would then capture the key unit-actions in the activity. And the graph edges $\mathcal{E}$ would capture the temporal relationship between these nodes (*i.e.* unit-actions).

Learning The Graph Nodes. In a dataset of human activities, unit-actions can be thought of as the dominant *latent* short-range concepts. That is, unit-actions are the building blocks of the human activity. So, in a graph-inspired representation of the activity, these unit-actions can act as the graph nodes $\mathcal{N}$. Assuming that it is prohibitively expensive to have unit-actions annotation for minutes-long videos, a challenge is how to represent them? In other words, how to represent the graph nodes? As a remedy, we opt for learning a set of N latent features Y, $Y = \{y_j \,|\, j = 1, 2, ..., N\}, Y \in \mathbb{R}^{N\times C}$. These features Y then become the vector representation of the graph nodes $\mathcal{N}$, *i.e.* $Y \equiv \mathcal{N}$.

A problem, however, is how to correlate each video feature x_i with each node in Y. To solve this, we propose the node attention block, inspired by self-attention block [21, 52, 48], shown in Fig. 3a. The node attention block takes as an input a feature x_i and all the node features Y. Then, it transforms the initial representation of the nodes from Y into $\hat{Y}$, using one hidden layer MLP with weight and bias $w \in \mathbb{R}^{C\times C}, b \in \mathbb{R}^{1\times C}$. This transformation makes the nodes learnable and better suited for the dataset inhand. Then, a dot product $\otimes$ is used to measure the similarity between x_i and $\hat{Y}$. An activation function σ is applied on the similarities to introduce non-linearity. The result is the activation values $\boldsymbol{\alpha} \in \mathbb{R}^{H\times W\times N}$. The last step is multiplying all the nodes $\hat{Y}$ with the activation values $\boldsymbol{\alpha}$, such that we attend to each node $\hat{y}_j$ by how much it is related to the feature x_i. Thus, the node attention block outputs the attended nodes $Z_i = \{z_{ij} \,|\, j = 1, 2, ..., N\}, Z_i \in \mathbb{R}^{N\times H\times W\times C}$. We refer to Z_i as node-attentive feature, and we refer to z_{ij} as the j-th node feature in Z_i. More formally,

$$\hat{Y} = w * Y + b \tag{1}$$

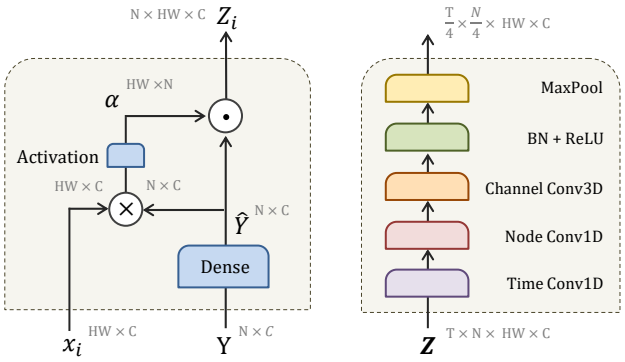
$$\boldsymbol{\alpha} = \sigma(x_i * \hat{Y}^T) \tag{2}$$

$$\begin{aligned} Z_i &= \boldsymbol{\alpha} \odot \hat{Y} \\ &= \alpha_j \odot y_j, \quad j = 1, 2, ..., N \end{aligned} \tag{3}$$

Hence, the vector representation of all video segments is a 5D tensor $\mathbf{Z} = \{Z_1, Z_2, ..., Z_T\}, \mathbf{Z} \in \mathbb{R}^{T\times N\times H\times W\times C}$. The

names of 5 dimensions in $\mathbf{Z}$ are: timesteps, nodes, width, height and channels. From now on, we use these 5 dimensions to express feature vectors and convolutional kernels.

In sum, the node attention block takes a feature x_i, corresponding to a short video segment s_i and measures how similar $\boldsymbol{\alpha}$ it is to learned set of latent concepts $\hat{Y}$. The similarities $\boldsymbol{\alpha}$ are then used to attend to the latent concepts. This is crucial for recognizing long-range videos, where the network is not feed-forwarded only with a short video segment x_i but with global representation Y. This gives the network the ability for focus on both local video signal x_i and global learned context $\hat{Y}$.

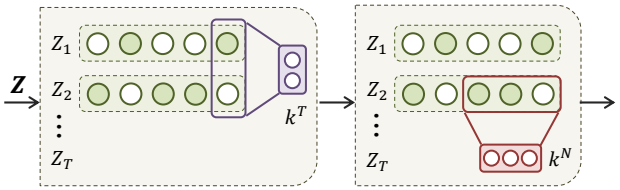


(a) Node Attention Block (b) Graph Embedding Layer

Figure 3: (a) Node attention block measures similarities $\boldsymbol{\alpha}$ between segment feature x_i and learned nodes $\hat{Y}$. Then, it attends to each node in $\hat{Y}$ using $\boldsymbol{\alpha}$. The result is the node-attentive feature Z_i expressing how similar each node to x_i. (b) Graph Embedding layer models a set of T successive node-attentive features $\mathbf{Z}$ using 3 types of convolutions. *i.* Timewise Conv1D learns the temporal transition between node-attentive features $\{Z_i, ..., Z_{i+t}\}$. *ii.* Nodewise Conv1D learns the relationships between nodes $\{z_{i,j}, ..., z_{i,j+n}\}$. *iii.* Channelwise Conv3D updates the representation for each node z_{ij}.

Our node attention block is different from the non-local counterpart [21] in twofold. First, the attention values are conditioned on local x_i and global $\hat{Y}$ signals. Second, non-local does tensor product between attention values $\boldsymbol{\alpha}$ and local signal x_i, while we attend by scalar multiplication between $\boldsymbol{\alpha}, \hat{Y}$ to retrain the node dimension. Lastly, our node attention block is much more simpler than the non-local, as we use only one fully-connected layer.

Learning The Graph Edges. Up till now, we have learned the graph nodes $\hat{Y}$. We have also represented each video segment s_i in terms of the nodes, as node-attentive feature Z_i. Next, we would like to learn the graph edges $\mathcal{E}$, and arrive at the final graph structure. To this end, we propose a novel graph embedding layer, shown in Fig. 3b. Regarding the graph edges, we are interested in two types of relationships. First, we are interested in the relationship between graph nodes. Loosely speaking, if nodes stand for unit-actions as "pour milk", "crack egg", we would like to learn how correlated are these two unit-actions when used in different activities as "make pancake" or "prepare coffee". Second, we are interested in how the graph nodes transition over time. For instance, we want to encode the significance of unit action "pour milk" comes after or before "crack egg" when it comes to recognizing "make pancake". Let's take t successive video segments $\{s_i, ..., s_{i+t}\}$. When processed by CNN and node attention block, they are represented as $\{Z_i, ..., Z_{i+t}\}$. To learn the temporal transition between them, we apply a one-dimensional convolution, (Conv1D) on the temporal dimension only. These timewise Conv1D, proposed by [2], are efficient in learning temporal concepts. One kernel learned by timewise Conv1D is the 5D tensor $k^T \in \mathbb{R}^{t \times 1 \times 1 \times 1 \times 1}$, where t is the kernel size. In total, we learn C kernels to keep the channel dimension of the features $\mathbf{Z}$ unchanged.



(a) Timewise Conv1D (b) Nodewise Conv1D

Figure 4: (a) Timewise Conv1D learns the temporal transition between successive nodes-embeddings $\{Z_i, ..., Z_{i+t}\}$ using kernel k^T of kernel size t. (b) Nodewise Conv1D learns the relationships between consecutive nodes $\{z_{i,j}, ..., z_{i,j+n}\}$ using kernel k^N of kernel size n.

Besides learning the temporal transition between node-attentive features $\{Z_i, ..., Z_{i+t}\}$, we also want to learn the relationship between the nodes themselves $\{z_{ij} \,|\, j = 1, 2, ..., N\}$ inside each node-attentive feature Z_i. The problem is that the adjacency matrix, which defines the graph structure, is unkown. A naive approach is to assume all nodes are connected. This leads to an explosion of N^2 edges – prohibitive to learn. To overcome this, we restrict the number of adjacents (*i.e.* neigbours) each node z_{ij} can have. In other words, we assume that each node z_{ij} is adjacent to only n other nodes. This makes it possible to learn edge weights using one-dimensional convolution, applied on only the node dimension of Z_i. We call this convolution nodewise Conv1D. One kernel learned by nodewise Conv1D is the 5D tensor $k^N \in \mathbb{R}^{1 \times n \times 1 \times 1 \times 1}$, where n is the kernel size. In sum, we learn C kernels to keep the channel dimension of the features $\mathbf{Z}$ unchanged.

Both timewise and nodewise Conv1D learn graph edges

separately for each channel in the features $\mathbf{Z}$. That is why we follow up with a typical spatial convolution (Conv2D) to model the cross-channel correlations in each node feature z_{ij}. Spatial Conv2D learns C different kernels, each is the 5D tensor $k^C \in \mathbb{R}^{1\times1\times1\times1\times C}$.

Having learned the graph edges using convolutional operations, we proceed with BatchNormalization and ReLU non-linearity. Finally, we downsample the entire graph representation $\mathbf{Z}$ over both time and node dimensions using MaxPooling operation. It uses kernel size 3 and stride 3 for both the time and node dimensions. Thus, after one layer of graph embedding, the result graph representation is reduced from $T\times N\times H\times W\times C$ to $(T/3)\times(N/3)\times H\times W\times C$.

4. Experiments

Implementation. When training VideoGraph on a video dataset, we uniformly sample $T = 64$ video segments from each video v. One segment s_i is a burst of 8 successive frames. When the 64 segments are fed-forward to I3D up to the last convolutional layer `res5_c`, the corresponding convolutional features for the entire video is $\mathbf{X} = \{x_i \,|\, i = 1, 2, ..., 64\}, \mathbf{X} \in \mathbb{R}^{64\times7\times7\times1024}$. We use $N = 128$ as the number of latent concepts. Both the video-level features $\mathbf{X}$ and latent concepts $Y \in \mathbb{R}^{128\times1024}$ are fed-forward to the node attention block. The result is the graph representation $\mathbf{Z} \in \mathbb{R}^{128\times64\times7\times7\times1024}$. Then, $\mathbf{Z}$ is passed to graph embedding layers to learn node edges and reduce the feature representation. In graph embedding layer, we use kernel size $t = 7$ for the timewise Conv1D and kernel size $n = 7$ for the nodewise Conv1D. In total, we use 2 successive layers of graph embedding. Their output feature is then feed-forwarded to a classifier to arrive at the vide-level predictions. The classifier uses 2 fully-connected layers with BatchNormalization and ReLU non-linearity. We use softmax as the final activation for single-label classification or sigmoid for multi-label classification.

VideoGraph is trained with batch-size 32 for 500 epoch. It is optimized with SGD with 0.1, 0.9 and 0.00001 as learning rate, momentum and weight decay, respectively. It is implemented using TensorFlow [58] and Keras [59].

4.1. Datasets

As this paper focus on human activities spanning many minutes, we choose to conduct our experiments on the following benchmarks: Breakfast [1], Epic-Kitchens [10] and Charades [11]. Other benchmarks for human activities contain short-range videos, *i.e.* a minute or less, thus do not fall within the scope of this paper.

Breakfast is a dataset for task-oriented human activities, with the focus on cooking. It is a video classification task of 12 categories of breakfast activities. It contains 1712 videos in total, 1357 for training and 335 for test. The average length of videos is 2.3 minutes. The activities are performed by 52 actors, 44 for training and 8 for test. Having different actors for training and test splits is a realistic setup for testing generalization. Each video is represents only one category of focus activity. Besides, each video has temporal annotation of unit-actions comprising the activity. In total, there are 48 classes of unit-actions. In our experiments, we only use the activity annotation, and we don't use the temporal annotation of unit-actions.

Epic-Kitchens is a recently introduced large-scale dataset for cooking activities. In total, it contains 274 videots performed by 28 actors in different kitchen setups. Each video represents a cooking different cooking activity. The average length of videos is 30 minutes, which makes it ideal for experimenting very long-range temporal modeling. Originally, the task proposed by the dataset is classification on short video snippets, with average length of ∼3.7 seconds. The provided labels are, therefore, the categories of objects, verbs and unit-actions in each video snippet. However, the dataset does no provide video-level category. That is why we consider all the object labels of a specific video as video-level label. Hence, posing the problem as multi-label classification of these videos. This setup is exactly the same used in Charades [11] for video classification. For performance evaluation, we use mean Average Precision (mAP), implemented in Sk-Learn [60].

Method	Modality	mAP (%)
Two-stream [17]	RGB + Flow	18.6
Two-stream + LSTM [17]	RGB + Flow	17.8
ActionVLAD [5]	RGB + iDT	21.0
Temporal Fields [17]	RGB + Flow	22.4
Temporal Relations [23]	RGB	25.2
ResNet-152 [61]	RGB	22.8
ResNet-152 + Timeception [2]	RGB	31.6
I3D [9]	RGB	32.9
I3D + ActionVLAD [5]	RGB	35.4
I3D + Timeception [2]	RGB	37.2
I3D + VideoGraph	RGB	**37.8**

Table 1: VideoGraph outperforms related works using the same backbone CNN. Results are for Charades dataset.

Charades is a dataset for multi-label classification of action videos. It consists of 8k, 1.2k and 2k video for training, validation and testing, respectively. is multi-label, action classification, video dataset with 157 classes. Each video spans 30 seconds and comprises of 6 unit-actions, on average. This is why we choose Charades, as it fits perfectly to the needs of this paper. For evaluation, we use mAP, as detailed in [11].

Method	Breakfast Acc. (%)	Breakfast mAP (%)	Epic-Kitchens mAP (%)
ResNet-152 [61]	41.13	32.65	–
ResNet-152 + ActionVLAD [5]	55.49	47.12	–
ResNet-152 + Timeception [2]	57.75	48.47	–
ResNet-152 + VideoGraph	**59.12**	**49.38**	–
I3D [9]	58.61	47.05	48.86
I3D + ActionVLAD [5]	65.48	60.20	51.45
I3D + Timeception [2]	67.07	61.82	**55.46**
I3D + VideoGraph	**69.45**	**63.14**	55.32

Table 2: VideoGraph outperforms related works using the same backbone CNN. We experiment 2 different backbones: I3D and ResNet-152. We experiment on two different tasks of Breakfast: single-label classification of activities and multi-label classification of unit-actions. And for Epic-Kitchens, we experiment on the multi-label classification.

4.2. Experiments on Benchmarks

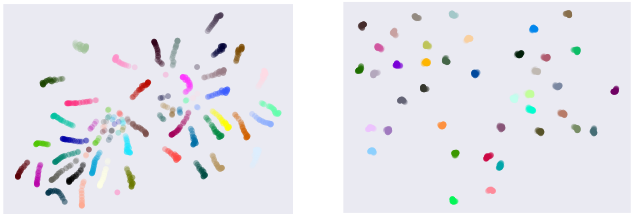
In this section, we experiment and evaluate VideoGraph on benchmark datasets: Breakfast, Charades and Epic-Kitchens, and we compare against related works. We choose two strong methods to compare against. The first is Timeception [2]. The reason is that it can model 1k timesteps, which is up to a minute-long video. Another reason is that Timeception is an order-ware temporal method. The second related work is ActionVLAD [5]. The reason is that it is a strong example of orderless method. It also can aggregate temporal signal for very long videos.

VideoGraph resides on top of backbone CNN, be it spatial 2D CNN, or spatio-temporal 3D CNN. So, in our comparison, we use two backbone CNNs, namely ResNet-152 [62] and I3D [9]. By default, I3D is designed to model a short video segment of 8 frames. But thanks to the fully-convolutional architecture, I3D can indeed process minutes-long video. This is made possible by average pooling the features of many videos snippets, in logit layer, *i.e.* before softmax activation [9]. ResNet-152 is a frame-level classifier. To extend it to video classification, we follow the same approach used in I3D and average pool the logits, *i.e.* before softmax. In all the following comparisons, we use 512 frames, or 64 segments, per video as input to I3D. And we use 64 frames per video as and input to ResNet-152.

Breakfast. Each video in this dataset depicts a complex breakfast activity. Thus, the task inhand is single-label classification. The evaluation metric used is the classification accuracy. We experiment our model on Breakfast, and we compare against baseline methods. The results are reported in table 2.

Epic-Kitchens.

When comparing VideoGraph against related works, see table 2, Timeception and VideoGraph, we notice that we are on bar with Timeception. VideoGraph performs better when trained on single-label video dataset, where each video has one label. This gives VideoGraph an ample opportunity to tailor the graph-inspired representation for each class. However, as mentioned, we pose the task in Epic-Kitchen as multi-label classification. That is, no single category for a video. That's when VideoGraph does not perform as good.



(a) (b)

Figure 5: Visualization of the learned graph nodes. In the first 20 epoch during training (left), VideoGraph updates the node features $\hat{Y}$ to increase the pairwise distance between them. That is, VideoGraph learns discriminant representations of the nodes. In the last 20 epoch during training (right), the learning cools down and barely their representation is updated. We visualize using t-SNE [63].

Charades. In this experiment, we evaluate our model on Charades dataset. And we compare the performance against recent works. The results are reported in Table 1. VideoGraph improves the performance of the backbone CNN. For VideoGraph, Charades is particularly challenging dataset, for two reasons. First, the average video length is 30 seconds, and VideoGraph learns better representstion for long-range videos. Second, it is a multi-label classification, and that's when VideoGraph is not able to learn category-specific unique graph.

4.3. Learned Graph Nodes

The proposed node attention block, see figure 4a, learns latent concept representation $\hat{Y}$ using fully-connected layer. This learning is conditioned on the initial value Y. We found that this initial value is crucial for VideoGraph to con-

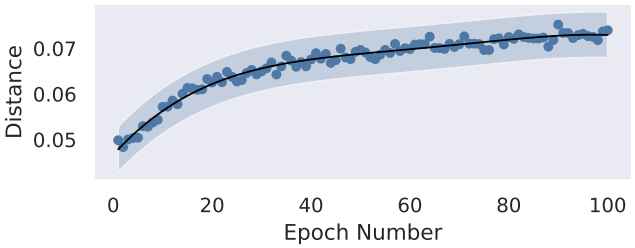


Figure 6: The pairwise Euclidean distances between normalized latent concepts $\hat{Y}$ increases rapidly in the beginning of the training, but it converges in the end.

verge. We experiment with 3 different types of initialization: *i.* random values, *ii.* Sobol sequence and *iii.* k-means centroids. Random values seems to be a natural choice, as all the learned weights in the model are randomly initialized before training. Sobol sequence is a plausible choice, as the sequence guarantees low discrepancies between the initial values. The last choice has proven to be successful in ActionVLAD [5]. The centroids are obtained by clustering the feature maps of the last convolutional layer of the backbone CNN. However, we do not find one winning strategy across the benchmarks used. We find that Sobol sequence is the best choice for training on Epic-Kitchens and Charades. While the random initialization gives the best results on Breakfast. In table 1, we report the performance of VideoGraph using different initialization choices for the latent concepts Y. In all cases, we see in figure 5 that the node attention layer successfully learns discriminant representations of latent concepts, as the training proceedes. In other words, the networks learns to increase the Euclidean distance between each pair of latent concepts. This is further demonstrated in figure 6.

Initialization	Epic-Kitchen mAP	Breakfast Acc.
Random	54.12	**69.45**
Sobol	**55.46**	65.61
K-means Centroids	52.47	—

Table 3: The initialization of the latent concepts is crucial for learning better representation $\hat{Y}$. We experimented with 3 choices: random, sobol, and k-mean clustering. Yet, there seems not to be one winning choice across different datasets.

4.4. Learned Graph Edges

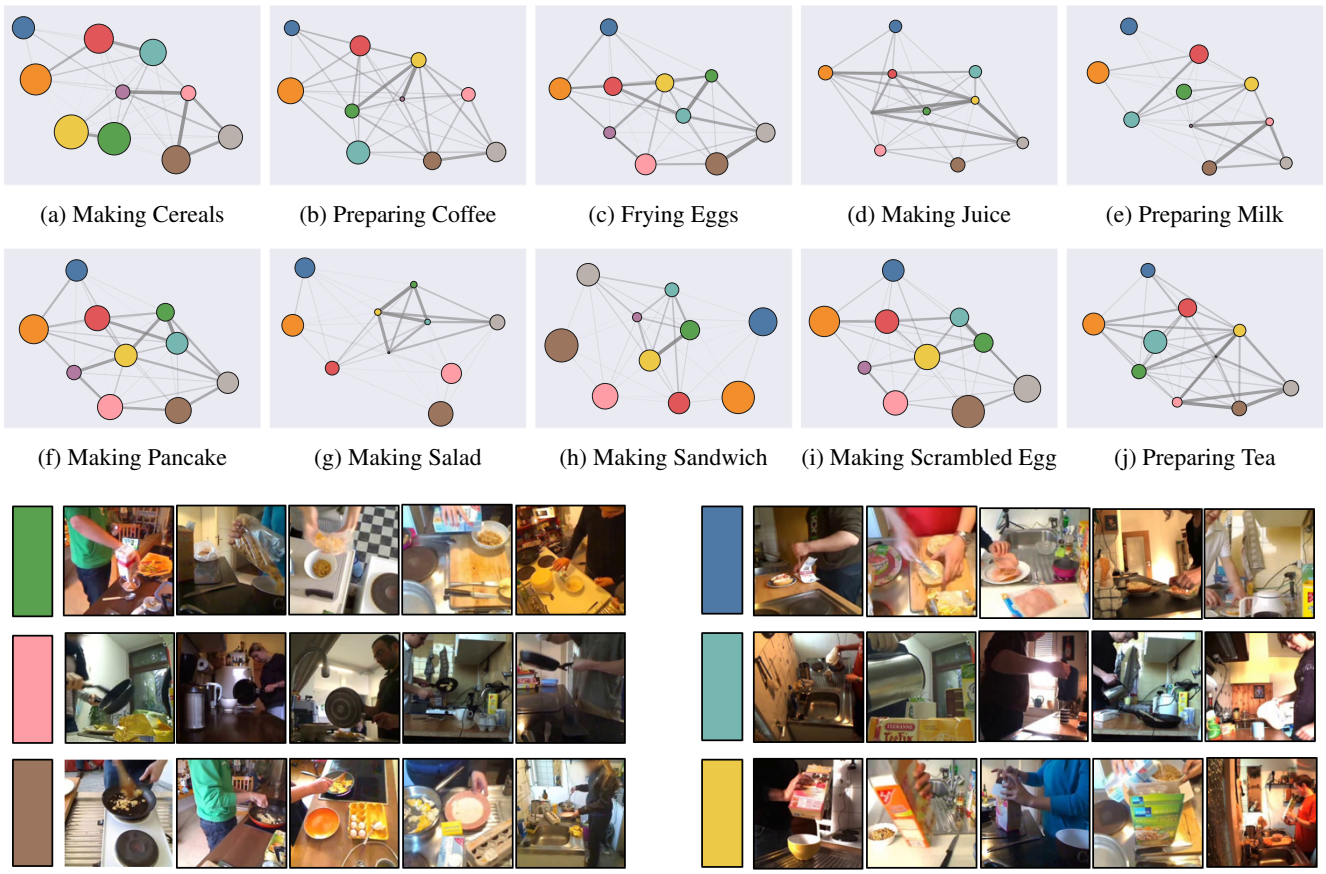
There are two types of graph edges, *i.e.* relationships, uncovered by VideoGraph. First, the timewise edges, *i.e.* how the nodes transition over time. Second, the nodewise edges, *i.e.* relationships between nodes themselves. To this end, we depend on the activation output of the second graph embedding layer. In other words, we extract the ReLU activation values. For M videos belonging to a specific human activity, the activation values are $z_1 \in \mathbb{R}^{M\times N\times T\times C}$, where C is the number of channels, T is the number of timesteps, and N is the number of nodes. First, we average the activations for all the videos, resulting in $z_2 \in \mathbb{R}^{N\times T\times C}$. Then, we average pool the activations over the temporal dimension, so we have $z_3 \in \mathbb{R}^{N\times C}$, summarizing the nodes representations for all the videos belonging to the specific activity. Finally, we measure the pairwise Euclidean distance between each pair in z_3. To plot the graph depicting the activity, we use these distances as the edge between the nodes. And to plot the nodes, we sum up the activations over the channel dimension in z_3. The result $z_4 \in \mathbb{R}^N$ is a scalar value reflecting the importance of the node to the activity. The graph is plotted using Fruchterman-Reingold force-directed algorithm, implemented in [64]. Figure 7 shows 10 different graph, each belonging to one human activity.

Importance of Temporal Structure. In this experiment, we validate by how much VideoGraph depends on the temporal structure and weak temporal order to recognize the human activities. To this end, we choose Breakfast, as it is temporally well-structured dataset. VideoGraph is trained on ordered set of 64 timesteps. We alter the temporal order of these timesteps and test the performance of VideoGraph. We use different alterations: *i.* random order, and *ii.* reversed order. Then, we measure the performance of VideoGraph, as well as baselines, on Breakfast testset.

Temporal Structure	Reversed (↓%)	Random (↓%)
I3D	0.0	0.0
I3D + ActionVLAD	0.0	0.0
I3D + Timeception	44.1	56.2
I3D + VideoGraph	22.5	55.9

Table 4: The drop of performance of VideoGraph and other models when changing the temporal order of the input video. Both VideoGraph and Timeceptions suffer huge drop in performace, as both are order-aware methods. On the other hand, ActionVLAD retains the same performance, as it is orderless method.

We notice, from table 4, a huge drop in performance for both VideoGraph and Timeception. However, as expected, no drop in performance for ActionVLAD, as it is completely orderless model. The conclusion is VideoGraph encodes the temporal structure of the human activities in breakfast. Added to this, it suffered slightly less drop in performance than Timeception. More importantly, figure 8 shows the confusion matrix of classifiyng the videos of Breakfast using two cases: *i* natural order of temporal video segments, and *ii.* random order of the video segments. We notice video graph makes more mistakes when trained on random order. It mistakes "scrambled egg" for "fried egg" if temporal order is neglected.



(a) Making Cereals (b) Preparing Coffee (c) Frying Eggs (d) Making Juice (e) Preparing Milk

(f) Making Pancake (g) Making Salad (h) Making Sandwich (i) Making Scrambled Egg (j) Preparing Tea

(k) Top related images to the nodes. These nodes are related to: cereal, pan, eggs, sandwich, kettle, and foodbox.

Figure 7: We visualize the relationship discovered by the first layer of graph embedding. Each sub-figure is related to one of the 10 activities in Breafast dataset. In each graph, the nodes represent the latent concepts learned by graph-attention block. Node size reflects how dominant the concept, while graph edges emphasize the relationship between these nodes.

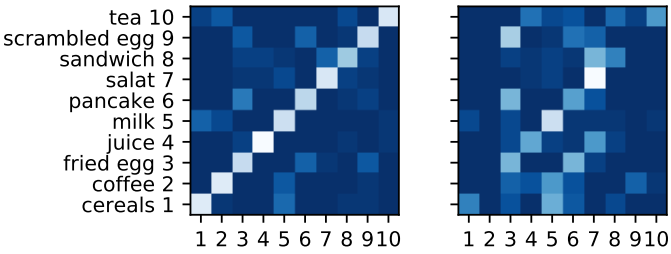


Figure 8: Confusion matrix for recognizing the 10 activity of Breakfast. VideoGraph is trained on random (right) *v.s.* correct temporal order (left). It mistakes "scrambled egg" for "fried egg" if temporal order is neglected.

5. Conclusion

To successfully recognize minutes-long human activities such as "preparing breakfast" or "cleaning the house", we argued that a successful solution needs to capture both the whole picture and attention to details. To this end, we proposed VideoGraph, a graph-inspired representation to model the temporal structure of such long-range human activities. Firstly, thanks to the node attention layer, VideoGraph can learn the graph nodes. This alleviate the need of node-level annotation, which is prohibitive and expensive in nowadays video dataset. Secondly, we proposed graph embedding layer. It learns the relationship between graph nodes and how these nodes transition over time. Also, it compresses the graph representation to be feed for a classifier. We demonstrated the effectiveness of VideoGraph on three benchmarks: Breakfast, Epic-Kitchens and Charades. VideoGraph achieves good performance on the three of them. We also discussed some of the upsides and downside of VideoGraph.

References

[1] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014.

[2] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Timeception for complex action recognition. In *arXiv*, 2018.

[3] Noureldien Hussein, Efstratios Gavves, and Arnold WM Smeulders. Unified embedding and metric learning for zero-exemplar event detection. In *CVPR*, 2017.

[4] Ionut C Duta, Bogdan Ionescu, Kiyoharu Aizawa, and Nicu Sebe. Spatio-temporal vlad encoding for human action recognition in videos. In *ICMM*, 2017.

[5] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell. Actionvlad: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.

[6] Amir Ghodrati, Efstratios Gavves, and Cees GM Snoek. Video time: Properties, encoders and evaluation. In *BMVC*, 2018.

[7] Gunnar A Sigurdsson, Olga Russakovsky, and Abhinav Gupta. What actions are needed for understanding human actions in videos? In *ICCV*, 2017.

[8] De-An Huang, Vignesh Ramanathan, Dhruv Mahajan, Lorenzo Torresani, Manohar Paluri, Li Fei-Fei, and Juan Carlos Niebles. What makes a video a video: Analyzing temporal information in video understanding models and datasets. In *CVPR*, 2018.

[9] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[10] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.

[11] Gunnar A Sigurdsson, Gúl Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016.

[12] Rohit Girdhar and Deva Ramanan. Attentional pooling for action recognition. In *NIPS*, 2017.

[13] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. In *arXiv*, 2017.

[14] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013.

[15] Xinyu Li, Yanyi Zhang, Jianyu Zhang, Shuhong Chen, Ivan Marsic, Richard A Farneth, and Randall S Burd. Concurrent activity recognition with multimodal cnn-lstm structure. In *arXiv*, 2017.

[16] 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 *CVPR*, 2015.

[17] Gunnar A Sigurdsson, Santosh Divvala, Ali Farhadi, and Abhinav Gupta. Asynchronous temporal fields for action recognition. In *CVPR*, 2017.

[18] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017.

[19] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018.

[20] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. In *arXiv*, 2017.

[21] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

[22] 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 *ECCV*, 2016.

[23] Bolei Zhou, Alex Andonian, and Antonio Torralba. Temporal relational reasoning in videos. In *arXiv*, 2017.

[24] Basura Fernando, Efstratios Gavves, José Oramas, Amir Ghodrati, and Tinne Tuytelaars. Rank pooling for action recognition. In *TPAMI*, 2017.

[25] 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. In *arXiv*, 2017.

[26] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011.

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

[28] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[29] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. In *arXiv*, 2016.

[30] Mathew Monfort, Bolei Zhou, Sarah Adel Bargal, Alex Andonian, Tom Yan, Kandan Ramakrishnan, Lisa Brown, Quanfu Fan, Dan Gutfruend, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. In *arXiv*, 2018.

[31] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The "something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

[32] Chunhui Gu, Chen Sun, Sudheendra Vijayanarasimhan, Caroline Pantofaru, David A Ross, George Toderici, Yeqing Li, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, et al. Ava: A video dataset of spatio-temporally localized atomic visual actions. In *arXiv*, 2017.

[33] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018.

[34] Marcus Rohrbach, Anna Rohrbach, Michaela Regneri, Sikandar Amin, Mykhaylo Andriluka, Manfred Pinkal, and Bernt Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. In *IJCV*, 2016.

[35] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *ACM PUC*, 2013.

[36] Guangnan Ye, Yitong Li, Hongliang Xu, Dong Liu, and Shih-Fu Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *ACM MM*, 2015.

[37] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. In *IJCV*, 2018.

[38] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011.

[39] Gunhee Kim and Eric P Xing. Reconstructing storyline graphs for image recommendation from web community photos. In *CVPR*, 2014.

[40] Bo Xiong, Gunhee Kim, and Leonid Sigal. Storyline representation of egocentric videos with an applications to story-based search. In *CVPR*, 2015.

[41] Jia-Yu Pan and Christos Faloutsos. Videograph: a new tool for video mining and classification. In *ACM/IEEE-CS DL*, 2001.

[42] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016.

[43] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[44] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[45] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Shuicheng Yan, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning network. In *arXiv*, 2018.

[46] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, 2018.

[47] Huang De-An, Shyamal Buch, Lucio Dery, Animesh Garg, Li Fei-Fei, and Juan Carlos Niebles. Finding "it": Weakly-supervised reference-aware visual grounding in instructional videos. In *CVPR*, 2018.

[48] Rohit Girdhar, Joäo Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *arXiv*, 2018.

[49] De-An Huang, Suraj Nair, Danfei Xu, Yuke Zhu, Animesh Garg, Li Fei-Fei, Silvio Savarese, and Juan Carlos Niebles. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *arXiv*, 2018.

[50] Shaojie Wang, Wentian Zhao, Ziyi Kou, and Chenliang Xu. How to make a blt sandwich? learning to reason towards understanding web instructional videos. In *arXiv*, 2018.

[51] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. In *ICLR*, 2017.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[53] Chao-Yuan Wu, Christoph Feichtenhofer, Haoqi Fan, Kaiming He, Philipp Krähenbühl, and Ross Girshick. Long-term feature banks for detailed video understanding. In *arXiv*, 2018.

[54] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[55] Zhenyang Li, Kirill Gavrilyuk, Efstratios Gavves, Mihir Jain, and Cees GM Snoek. Videolstm convolves, attends and flows for action recognition. *CVIU*, 2018.

[56] Yang Du, Chunfeng Yuan, Bing Li, Lili Zhao, Yangxi Li, and Weiming Hu. Interaction-aware spatio-temporal pyramid attention networks for action classification. In *ECCV*, 2018.

[57] Zhengyuan Yang, Yuncheng Li, Jianchao Yang, and Jiebo Luo. Action recognition with spatio-temporal visual attention on skeleton image sequences. In *IEEE ToCS*, 2018.

[58] Martín Abadi et al. Tensorflow. tensorflow.org, 2015.

[59] François Chollet et al. Keras. keras.io, 2015.

[60] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 2011.

[61] Charades algorithms. github.com/gsig/charades-algorithms, 2017.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[63] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 2008.

[64] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In *SciPy*, 2008.