VidSage: Unsupervised Video Representational Learning with Graph Convolutional Networks

Ali Aminian

Stanford University
aaminian@stanford.edu

Abstract. Video understanding plays a vital role in several real-world applications such as surveillance systems, and autonomous vehicles, with a significant impact on the abilities of the agents to perform proper tasks. Yet, it is a challenging task given the lack of sufficient labeled data, its computational cost due to the added temporal dimension, and the challenging nature of conceptually understanding a video.
In this work, we proposed "VidSage", a system that transforms the input video into a generic representation in an unsupervised and self-supervised fashion. The model encodes the video both visually and semantically. Visual features are captured by a pre-trained image-level feature extractor. Graph neural networks are leveraged to capture semantic similarities between videos based on the appearing objects. Additionally, we proposed LiteVidSage, a fast and efficient alternative of VidSage, with improved runtime for on-device and offline applications.
Finally, we compared our model against prior works on standard benchmarks. We achieved 42.5% accuracy on the YouTube-8m dataset outperforming previous methods by absolute 9%. Moreover, we obtained 54% and 28% classification accuracy on Charade, and Moments in Time datasets, outperforming previous unsupervised methods by 9% and 17% respectively, and on par with the recent meta-learning-based work by Google [1].
The code is publicly available at https://github.com/aliiaminian.

Keywords: graph convolutional networks, video understanding, representational learning, self-supervised learning

1 Introduction

Video understanding is a very challenging problem with a significant impact on many applications. For instance, action recognition can be leveraged in surveillance systems, robotics, and human-computer interaction. Understanding the context of a video leads to several benefits such as video summarization systems, and highlight extraction applications, capturing not only the interesting moments but also maintaining the consistency across the video.

However, there are several challenges with videos. First, the temporal dimension of the video requires intense computations and sufficient memory, which is

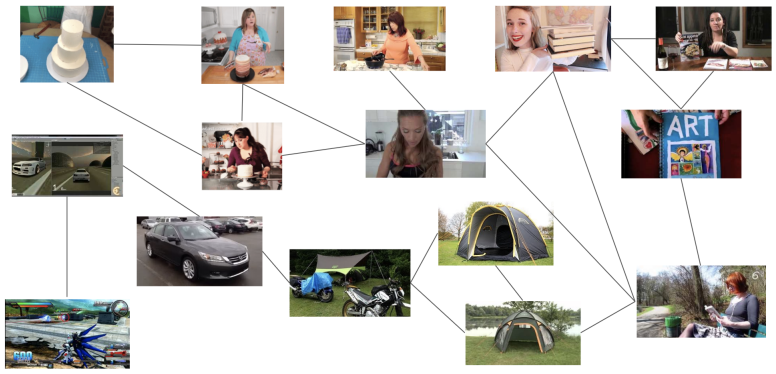

Fig. 1. VidSage Graph. Each image represents a sample frame from the video. Edges have normalized weight based on the number of in common objects between the videos.

infeasible to operate on large scale data. Moreover, labeled datasets are limited, and manual labeling of the videos is not cost-effective. Finally, not only visually understanding the frames is important, but also conceptually understanding the video is critical, which the latter is very challenging for machines. Therefore, learning generic representations of videos will significantly alleviate the computation costs and lead to better downstream models.

With the recent advances in convolutional neural networks (CNNs), and Graph Convolutional Networks (GCNs), many attempts have been made to learn video actions either by CNNs and sequential models such as LSTM [2], or GCNs such as [3] by extracting person's joints as the key-points across frames, and predicting the action from the produced graph. However, both have their limitations. They ignore either the topological structures between data points, or the appearance of the data points.

In this work, we proposed a model that both visually and semantically encodes the video by modeling the conceptual relations between videos by a GCN. More specifically, we build a graph G with every video as a node. We form different types of connections between nodes based on the semantic similarity of the videos. First, we connect visually similar videos in the graph by computing average frame-level features using a pre-trained image feature extractor. Second, we identify objects in the videos as a self-supervisory signal and use it as another notion of similarity between videos. Note that these are valuable implicit information in the videos which do not require prior labels and costs. Once the graph is built, we follow [4] and [5] to iteratively transform features and propagate information across neighbors to capture k-hops away structures in the graph per video. This transformation leads to a richer video-level representation, with a deeper understanding of the video, which is learned in a self-supervised fashion.

To measure the richness of the representations, we use the video-level features and train a simple supervised neural net for action recognition, and video classification. Different variations of Graph Neural Networks (GNN), including GCN,

GraphSage, and Graph Attention Network (GAT), were explored in this work. We compared our model against prior state-of-the-art works, and we achieved superior results. We outperformed YouTube-8m [6] video classification benchmark by 9%. Additionally, we outperformed recent meta-learning-based work by Google [7] on action recognition and improved the classification mAP by 19%. Compared to unsupervised works, we significantly outperformed all previous methods. To the best of our knowledge, this is the first work that combines both appearance modeling to capture visual features, and GCN variants to propagate contextual information and capture semantics of the video.

2 Related works

In this section, we discuss prior work on video understanding, GCNs, and attempts to leverage GCN for video representation learning.

2.1 Video Understanding

With the recent advances in convolutional neural networks (CNNs) and sequence models, such as GRUs [8] and Long Short Term Memory (LSTM)[2], many attempts have been made to learn video features. [9] used a many-to-one LSTM model to understand the relations between frames. Bidirectional LSTM used to process the relations from both directions. On the other hand, [10] used C3d and [11] used I3D to learn video representation with 3D convolutions. However, these methods are computationally costly and have limitations to parallelize the computations. Additionally, we have vanishing gradient problem for long videos, causing challenges to learn long-term dependencies. Later on, [12] proposed attention mechanism, and was used by [13] to address long-term dependencies by attending to relevant features in videos. However, the computation cost is still problematic. Afterward, [14] proposed transformers, which models pairwise dependencies in parallel, and [15] used for action recognition and video classification applications. Nevertheless, most of the methods are relying on labeled data.

Action Recognition. Traditionally, two networks were used to capture visual information and motion, followed by different fusing techniques to aggregate the results[16]. More recently, [17] proposed a slow-fast network with superior results on standard benchmarks for action classification. Google recently announced AssembleNet model [1], which uses meta-learning to find better architectures for action classification.

Video Classification. [6] released YouTube-8m last year, which is 8 million videos with 3800 video classes. The classes are not necessarily actions and represent the events that occurred in the video.

2.2 Graph Neural Networks

On the other hand, GNNs and Graph Convolutional Networks (GCNs) has gained significant attention recently, by capturing the topology in the networks

and learning the structures. More traditionally, spectral methods were used to capture the structure of the networks [18]. Later, [3] proposed GCNs that outperform traditional methods. [5] proposed GraphSage, which allows inductive inferences at large scale networks with millions of nodes and billions of edges. More recently, [4] proposed Graph Attention Networks, which incorporates the attention mechanism in graphs. [19] used a GCN to model a person's joints as the key-points across frames in order to perform activity recognition.

3 Method

3.1 Terminology

In a GCN, we have a graph G, with n nodes and e edges. Each node N_i has an initial feature $x_i = [x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,d}]$ of dimension d, and feature matrix $X \in \mathbb{R}^n \times \mathbb{R}^d$. We keep the graph information in an adjacency matrix A.

Every neural network layer can then be written as:

$$H^{l+1} = f(H^l, A)$$

Initially, $H^0 = X$, and after the last layer, we have $H^L = Z$. L denotes the total number of layers in the GCN. Choosing and parameterizing function f will lead to different variants of GCNs. Matrix $Z \in \mathbb{R}^n \times \mathbb{R}^d$ has the final embeddings per node. New features are iteratively computed for each node by keeping valuable information from its previous representation, as well as capturing the node's local and global structure in the graph.

For every node, embeddings are computed at layer t as:

$$h_i^t = \varphi(h_i^{t-1}, AGGR(\phi(N(i))$$

$$N(i) = (\forall u : A_{i,u} = 1)$$

ϕ and φ are arbitrary transformations, mainly single layer neural network with parameter W_1 and W_2, followed by non-linearity and dropout. $AGGR$ can be replaced by different neighborhood aggregation functions such as max-pooling, mean, or sum.

A schematic of this architecture can be seen in Figure 2.

3.2 Preprocessing

The goal is to compute initial representative features per video. To compute, we follow [6] for fair-comparison and use frame-level feature extraction from the Inception model [20] pre-trained on ImageNet [21], and finally aggregate frame features. Formally, a video-level feature for video ν: $f_\nu = \varphi(x^\nu_{1:F})$ is a fixed-length representation at the video-level. $x_i^v \in \mathbb{R}^{1024}$ denotes frame-level Inception features of frame i in video ν. φ is an arbitrary aggregation function such as a sequential model, or simple averaging. $f_\nu \in \mathbb{R}^{1024}$ denotes video-level features of the video ν.

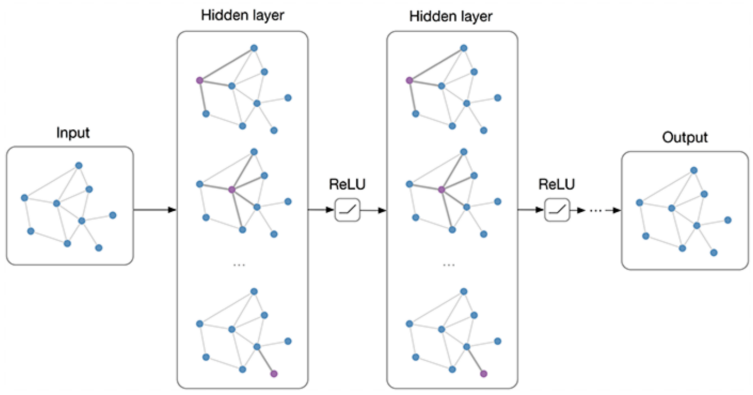

Fig. 2. A GCN with two convolutional layers and a ReLU non-linearities.

In this work, we explored simple averaging, as well as sequential models such as LSTM [2], to compute video feature-vector from the frame-level features. However, the gain is not significant in sequential models, given the added complexity and latency. Therefore, we continued with the averaging technique followed by normalization in the rest of this work and demonstrated that despite its simplicity, it outperforms previous intricate designs. This is visualized in Figure 3.

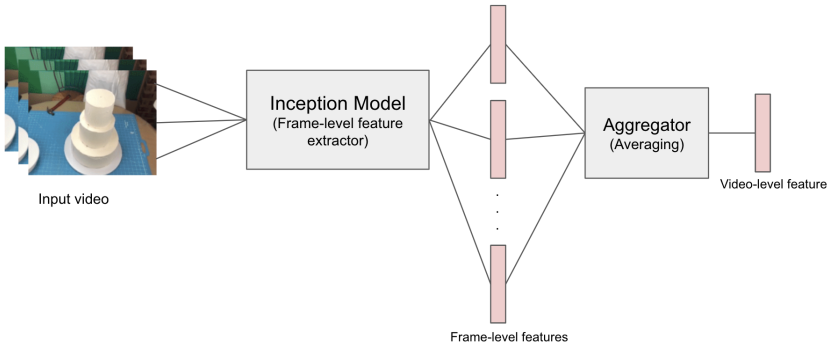

Fig. 3. Obtaining video-level features from a given video.

3.3 Model Architecture

We proposed two unsupervised models for video representation learning. LiteVidSage, our efficient alternative with fast inference, and VidSage, which leverages

objects in the videos to understand the context more deeply, and produce video-level features accordingly.

We model the semantic relations between the videos by graphs. For a given set of n videos $(\nu_1, \nu_2, ..., \nu_n)$, we use a GCN to model the relations and propagate information from each video to its conceptually similar videos.

Formally, we build a graph G, where every video ν_i is a node in the graph. Nodes u and v are connected if $dis(u, v) > \tau$ where $dis : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ measures the similarity between u and v, and τ is a threshold factor to control the sparsity of the graph G. For every node u, we compute initial video-level features to get $f_u \in \mathbb{R}^d$. In our GCN settings, we have $L = 2$, with ϕ and φ as single layer perceptron with weights $W^1 \in \mathbb{R}^{d \times d}$ and $W^2 \in \mathbb{R}^{d' \times d}$, followed by ReLU, and a dropout layer.

At every layer t, we compute new embeddings by:

$$h_i^t = ReLU(W_2 | h_i^{t-1}, \sum_{j \in N(i)} ReLU(W_1 h_j^{t-1}) |)$$

where $h_i^0 = f_\nu \in \mathbb{R}^{1024}$ and $z_i = h_i^L \in \mathbb{R}^{d'}$. $||$ denotes concatenation. Note that the weights W_1 and W_2 are shared across nodes and layers.

We normalize the node embedding at every iteration after aggregation and transformations of all the nodes. We perform neighbor sampling as [5] to enhance performance and its inductive inference feature, which is very desirable for learning generic representations for unseen videos.

Additionally, we explored GCN with self-attention mechanism following [4], by training a one layer MLP $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^1$ to produce the importance of nodes. Therefore:

$$e_{ij}^l = f(h_i^l, h_j^l)$$

To make coefficients easily comparable across nodes, we apply softmax function to the attention values. This can be seen in Figure 4.

$$a_{ij}^l = softmax_j(e_{ij}^l) = \frac{exp(e_{ij}^l)}{\sum_{k \in N_i} exp(e_{ik}^l)}$$

We define the objective function similar to [5]:

$$J_G(z_u) = -log(\sigma(z_u^T z_v)) - Q.E_{v_n \sim P_n(v)} log(\sigma(-z_u^T z_v))$$

Triplet loss is used by creating triplets of query node q, a random neighbor of q called pos, and a non-adjacent node neg, and using a similarity metric to bring adjacent nodes closer in the embedding space, and push negative pairs away from each other. More formally:

$$J_G(i) = max(dis(q_i, pos_i) - dis(q_i, neg_i) + margin, 0)$$

where i denotes i^{th} triplet sample in our dataset. We also explored contrastive loss and obtained similar results.

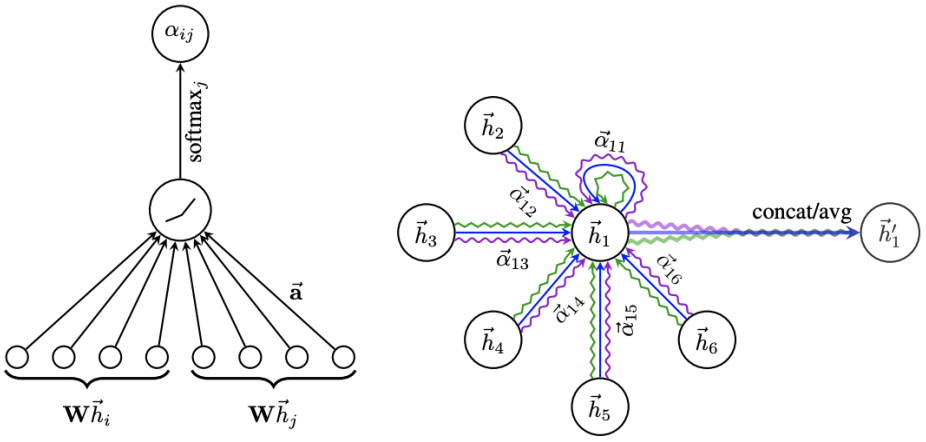

Fig. 4. A Graph Attention Mechanism with multiple heads. This figure is borrowed from [4].

LiteVidSage. We define the similarity between two videos ν_i and ν_j, with initial video-level features f_i and f_j based on their Euclidean distance. Two nodes u and v are connected if and only if:

$$g(u,v) = 1 - \sqrt{\sum_{j=1}^{d}(x_u^j - x_v^j)^2} \geq \tau$$

Please note that since the features are normalized, this would be proportional to the cosine similarity. This notion of similarity will lead to a tighter clusters of classes when trained, with better boundaries across different classes. As a consequence, transformed embeddings $Z \in \mathbb{R}^{d'}$ are closer for visually similar videos.

VidSage. As opposed to the LiteVidSage, not only we form connections between visually similar videos, but also we incorporate video semantics to build the graph.

For every video ν_u, we uniformly sample frames with sample rate r, and run an off-the-shelf object detection on the sampled frames, and finally aggregate the detected objects. This will lead to $O_u = \{o_{u,1}, o_{u,2}, ..., o_{u,k}\}$, where o_i is an object entity that appeared in the video u. We form a weighted edge between nodes u and v by:

$$w_{uv}^1 = \frac{|O_u \cap O_v|}{|O_u \cup O_v|}$$

Additionally, we connect similar nodes in Euclidean space as before, with weight one:

$$w_{uv}^2 = \mathbb{1}(Euclidean(u,v) > \tau)$$

Finally, we use α and β to control the contribution of visual and semantic connections by:

$$w_{uv} = \alpha w^1_{uv} + \beta w^2_{uv}$$

In this work, we experimented with $\alpha = 0.7$ and $\beta = 0.3$. Note that running object detection takes 0.05 seconds on average on a single V100 GPU for a video of length 15 seconds. Thus we can feasibly run this during the inference. Additionally, we use r to control this latency.

3.4 Training

We train our model with two layers to avoid the over-smoothing effect, along with neighborhood sampling of sizes $S1 = 25$, and $S2 = 10$ per layer. We use rectified linear units (ReLU) as our non-linearity, and a batch size of size 512. We train on a single Tesla V100 GPU, for 12 hours. We use a cosine decay for the learning rate with a maximum value of 10^{-5} and a linear warm-up for 10 steps. We apply dropout with a probability of 0.5 after each transformation.

3.5 Inference

For a given unseen video v, we perform pre-processing step to obtain initial video features. Then, we apply an object detector on uniformly sampled frames and uniquely aggregate them into one list. Afterward, we connect the video's node to its nearest neighbors, and also to the nodes with in-common objects with proper weights, normalized by α and β. Finally, VidSage transforms the initial features f_v and produces final embedding z_v.

4 Experiments

In this section, we evaluate the learned representations by training a classifier on two video understanding downstream tasks: action recognition, and video classification. Next, we present our experiments and evaluation results.

4.1 Evaluation Metrics

Mean Average Precision(mAP): For every class, we bucketize the scores according to the model's prediction. At a given threshold τ, the precision $P(\tau)$ and recall $R(\tau)$ are given by:

$$P(\tau) = \frac{\sum_{t \in T} \mathbb{1}(y_t > \tau) g(t)}{\sum_{t \in T} \mathbb{1}(y_t > \tau)}$$

$$R(\tau) = \frac{\sum_{t \in T} \mathbb{1}(y_t > \tau) g(t)}{\sum_{t \in T} g(t)}$$

where $\mathbb{1}(.)$ is an indicator function. Average precision approximates the area under the curve of precision-recall by:

$$AP = \sum_{j=1}^{D} P(\tau_j)[R(\tau_j) - R(\tau_{j+1})]$$

where D indicates the number of buckets in our dataset. Finally, mean average precision is the mean of per-class AP.

$$mAP = \frac{\sum_{c=1}^{C} AP_c}{C}$$

where C denotes the total number of classes.

Top-k accuracy. This metric measures the fraction of samples that are correctly classified in the top-k predictions.

4.2 Action Recognition

We use YouTube-8m as the benchmark for video classification. It is a large-scale video classification dataset with over 35000 hours of videos, 2.6 billion pre-computed features, and 3862 classes. We compare our proposed model with state-of-the-art methods on YouTube-8m, such as [6]. In table 2, we demonstrated different variants of the VidSage network, using a pre-trained VidSage model and fine-tuned with a classification layer. Additionally, as it can be seen from table 1, we gain 2x latency improvement in LiteVidSage, with accuracy loss on some tasks. YouTube-8m's class and entity distribution can be seen in Figure 5.

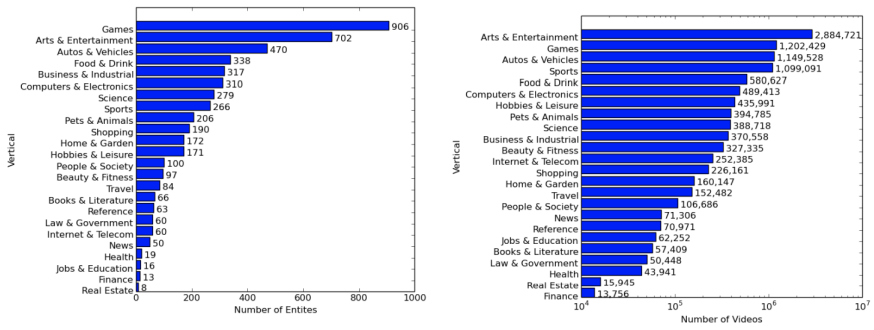

Fig. 5. Youtube-8m class and entity distributions according to [6].

4.3 Activity Recognition

Moments in Time dataset. The Moments in Time (MiT) dataset [22] is a large-scale video classification dataset with more than 800K videos (about 3

Table 1. Accuracy vs. Latency in proposed methods. Latency in seconds indicates the amount of time needed to generate a video representation based on a 5 seconds long video. It is computed on a single V100 GPU machine. Accuracy is the average of mAP in the three explored datasets.

Method	mAP	Latency(s)
VidSage	41.8	4.1
LiteVidSage	32.9	2.3

Table 2. Reported state-of-the-art video classification performances on YouTube-8m dataset. Methods as also reported in [6], contains video-level and frame-level features as described in [6]. refer to [6] for Precision at equal recall rate (PERR).

Input features	Method	mAP	Top-1 accuracy	PERR
Frame-level, $\{x^v_{1:F_v}\}$	Logistic + Average	11.0	50.8	42.2
Frame-level, $\{x^v_{1:F_v}\}$	Deep Bag of Frames	26.9	62.7	55.1
Frame-level, $\{x^v_{1:F_v}\}$	LSTM	26.6	64.5	57.3
Video-level, μ	Hinge loss	17.0	56.3	47.9
Video-level, μ	Logistic Regression	28.1	60.5	53.0
Video-level, μ	Mixture-of-2-Experts	29.6	62.3	54.9
Video-level, $[\mu; \sigma; Top-5]$	Mixture-of-2-Experts	30.0	63.3	55.8
Video-level	LiteVidSage (Ours)	33.1	65.2	–
Video-level	VidSage (Ours)	**42.5**	**73.0**	–

seconds per video). According to [7], it is a very challenging dataset with the state-of-the-art models obtaining less than 30% Top-1 accuracy. Our method outperforms most of the supervised state-of-the-art and all of the unsupervised methods. Results can be observed in Table 3.

Table 3. Reported state-of-the-art action classification performances on Moments in Time[22]

Method	Modality	Top-1 accuracy	Top-5 accuracy
BatchNorm[23]	Flow	11.60	27.40
TSN-2Stream[24]	RGB+F	25.32	50.10
ResNet50-ImageNet	RGB	27.16	51.68
Two-stream (2+1)D ResNet-50	RGB+F	28.97	55.55
I3D [11]	RGB+F	29.51	56.6
AssembleNet-50[1]	RGB+F	33.91	60.86
AssembleNet-101 [1]	RGB+F	**34.27**	**62.71**
LiteVidSage	RGB	18.9	39.9
VidSage	RGB	28.4	51.8

Charades. We also test on the popular Charades dataset [25], which is unique in the activity recognition domain as it contains long sequences. It is a large-scale dataset with 800k training examples and 33900 validation examples with 339 activity classes. Our results, along with other methods, can be seen in Table 4.

Table 4. Reported state-of-the-art action classification performances on Charades [25]

Method	pre-train	modality	mAP
2-stream [26]	UCF101	RGB+Flow	18.6
Asyn-TF [27]	UCF101	RGB+Flow	22.4
CoViAR[28]	ImageNet	Compressed	21.9
I3D [11]	Kinetics	RGB	32.9
I3D-NL [11]	Kinetics	RGB	37.5
SlowFast[17]	Kinetics	RGB+Flow	45.2
2-stream [1]	ResNet-50	RGB+Flow	48.7.6
2-stream (2+1)D [1]	ResNet-101	RGB+Flow	50.6
AssembleNet-50[1]	Kinetics	RGB+Flow	56.6
AssembleNet-101 [1]	Kinetics	RGB+Flow	**58.6**
VidSage (Ours)	YT-8m	RGB	54.1
LiteVidSage (Ours)	YT-8m	RGB	44.8

5 Conclusions

A good representation of videos is still unexplored due to limited labeled data, computation cost, and challenges of semantically understanding the video. In this work, we proposed VidSage, capturing both visual and contextual information of the video in a self-supervised way. A pre-trained image-level feature extractor was used to capture the video visually, and graph convolutional networks were leveraged to model the semantic relations between videos. We evaluated our model on two downstream tasks: action recognition and video classification. We outperformed previous works, which confirms the richness of learned representations. As the future work, we intend to represent each video with a graph, by modeling the relations between frames, and a pooling mechanism in order to get a feature vector per video. This direction is not explored yet and may lead to finding useful semantic relations between frames.

References

1. Ryoo, M., Piergiovanni, A., Tan, M., Angelova, A.: Assemblenet: Searching for multi-stream neural connectivity in video architectures. (05 2019)
2. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8) (November 1997) 1735–1780
3. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
4. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. International Conference on Learning Representations (2018)
5. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS. (2017)
6. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, A.P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A large-scale video classification benchmark. In: arXiv:1609.08675. (2016)
7. Piergiovanni, AJ; Angelova, A.R.M.S.: Tiny video networks. ArXiv **2019arXiv191006961P** (2019)
8. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling (2014) cite arxiv:1412.3555Comment: Presented in NIPS 2014 Deep Learning and Representation Learning Workshop.
9. Li, C., Wang, P., Wang, S., Hou, Y., Li, W.: Skeleton-based action recognition using lstm and cnn. 2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW) (2017) 585–590
10. Tran, D., Bourdev, L.D., Fergus, R., Torresani, L., Paluri, M.: C3d: Generic features for video analysis. ArXiv **abs/1412.0767** (2014)
11. : Quo vadis, action recognition? a new model and the kinetics dataset. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 4724–4733
12. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR **abs/1409.0473** (2014)
13. Rahman, T., Rochan, M., Wang, Y.: Convolutional temporal attention model for video-based person re-identification. (04 2019)
14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17, USA, Curran Associates Inc. (2017) 6000–6010
15. Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: Video action transformer network. In: CVPR. (2018)
16. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classication with convolutional neural networks. In: Proceedings of International Computer Vision and Pattern Recognition (CVPR 2014). (2014)
17. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. ArXiv **abs/1812.03982** (2018)
18. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic. NIPS'01, Cambridge, MA, USA, MIT Press (2001) 849–856
19. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: AAAI. (2018)

20. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR). (2015)
21. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09. (2009)
22. Monfort, M., Andonian, A., Zhou, B., Ramakrishnan, K., Bargal, S.A., Yan, T., Brown, L., Fan, Q., Gutfruend, D., Vondrick, C., et al.: Moments in time dataset: one million videos for event understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019) 1–8
23. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, JMLR.org (2015) 448–456
24. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. Volume 9912. (10 2016)
25. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: Crowdsourcing data collection for activity understanding. In: European Conference on Computer Vision. (2016)
26. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., eds.: Advances in Neural Information Processing Systems 27. Curran Associates, Inc. (2014) 568–576
27. Sigurdsson, G., Divvala, S., Farhadi, A., Gupta, A.: Asynchronous temporal fields for action recognition. (12 2016)
28. Cao, H., Yu, S., Feng, J.: Compressed video action recognition with refined motion vector. (10 2019)