GraphVid: It Only Takes a Few Nodes to Understand a Video

Eitan Kosman and Dotan Di Castro

Bosch Center of AI, Haifa, Israel
{Eitan.Kosman,Dotan.DiCastro}@bosch.com

Abstract. We propose a concise representation of videos that encode perceptually meaningful features into graphs. With this representation, we aim to leverage the large amount of redundancies in videos and save computations. First, we construct superpixel-based graph representations of videos by considering superpixels as graph nodes and create spatial and temporal connections between adjacent superpixels. Then, we leverage Graph Convolutional Networks to process this representation and predict the desired output. As a result, we are able to train models with much fewer parameters, which translates into short training periods and a reduction in computation resource requirements. A comprehensive experimental study on the publicly available datasets Kinetics-400 and Charades shows that the proposed method is highly cost-effective and uses limited commodity hardware during training and inference. **It reduces the computational requirements 10-fold** while achieving results that are comparable to state-of-the-art methods. We believe that the proposed approach is a promising direction that could open the door to solving video understanding more efficiently and enable more resource limited users to thrive in this research field.

1 Introduction

The field of video understanding has gained prominence thanks to the rising popularity of videos, which has become the most common form of data on the web. On each new uploaded video, a variety of tasks can be performed, such as tagging [18], human action recognition [37], anomaly detection [46], etc. New video-processing algorithms are continuously being developed to automatically organize the web through the flawless accomplishment of the aforementioned tasks.

Nowadays, Deep Neural Networks are the de-facto standard for video understanding [35]. However, with every addition of a new element to the training set (that is, a full training video), more resources are required in order to satisfy the enormous computational needs. On the one hand, the exponential increment in the amount of data raises concerns regarding our ability to handle it in the future. On the other hand, it has also spurred an highly creative research field aimed at finding ways to mitigate this burden.

Among the first-generation of video processing methods were ones geared toward adopting 2D convolution neural networks (CNNs), due to their computational efficiency [43]. Others decomposed 3D convolutions [14,56] into simpler operators, or split a complex neural network into an ensemble of lightweight networks [9]. However, video understanding has greatly evolved since then, with the current state-of-the-art methods

featuring costly attention mechanisms [4,20,31,3,15,6,30]. Beyond accuracy, a prominent advantage of the latest generation of methods is that they process raw data, that is, video frames that do not undergo any advanced pre-processing. Meanwhile, pursuing new video representations and incorporating pre-computed features to accelerate training is a promising direction that requires more extensive research.

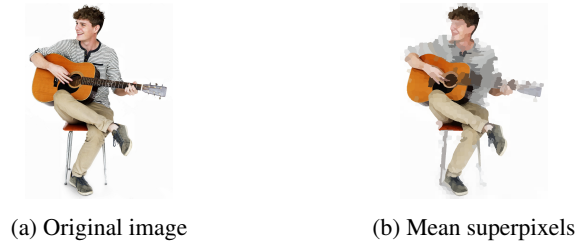
(a) Original image   (b) Mean superpixels

Fig. 1: A visual comparison between a pixel and a mean-superpixel representation. On the left, the original image is presented. On the right, we present the image formed by generating superpixel regions using SLIC and filling each region with its mean color.

Prior to the renaissance of deep learning [29], much research was done on visual feature generation. Two prominent visual feature generation methods are superpixels[1] and optic-flow[2]. These techniques' ability to encode perceptually meaningful features has greatly contributed to the success of computer vision algorithms. Superpixels provide a convenient, compact representation of images that can be very useful for computationally demanding problems, while optic-flow provides hints about motion. We rely on these methods to construct a novel representation of videos that encodes sufficient information for video understanding: 1) adjacent pixels are grouped together in the form of superpixels, and 2) temporal relations and proximities are expressed via graph connectivity. The example depicted in Figure 1 provides an intuition for the sufficiency of superpixel representation for scene understanding. It contains the superpixel regions obtained via SLIC [2], with each region filled with the mean color. One can clearly discern a person playing a guitar in both images. A different way of depicting the relations between superpixels is a graph with nodes representing superpixels [33,11,5]. Such a representation has the advantage of being invariant to rotations and flips, which obviates the need for further augmentations. We here demonstrate how this representation can reduce the computational requirements for processing videos.

Recent years have seen a surge in the utilization of Graph Neural Networks (GNNs) [26] in tasks that involve images [33,11,5], audio [12,61] and other data forms [54,55,1]. In this paper, we propose *GraphVid*, a concise graph representation of videos that enables video processing via GNNs. *GraphVid* constructs a graph representation of videos

---

[1] Superpixel techniques segment an image into regions by considering similarity measures, defined using perceptual features.

[2] Optic-flow is the pattern of the apparent motion of an object(s) in the image between two consecutive frames due to the movement of the object or the camera.

that is subsequently processed via a GCN to predict a target. We intend to exploit the power of graphs for efficient video processing. To the best of our knowledge, we are the first to utilize a graph-based representation of videos for efficiency. *GraphVid* dramatically reduces the memory footprint of a model, enabling large batch-sizes that translate to better generalization. Moreover, it utilizes models with an order-of-magnitude fewer parameters than the current state-of-the-art models while preserving the predictive power. **In summary, our contributions are:**

1. We present *GraphVid* - a simple and intuitive, yet sufficient representation of video clips. This simplicity is crucial for delivering efficiency.
2. We propose a dedicated GNN for processing the proposed representation. The proposed architecture is compared with conventional GNN models in order to demonstrate the importance of each component of *GraphVid*.
3. We present 4 types of new augmentations that are directly applied to the video-graph representation. A thorough ablation study of their configurations is preformed in order to demonstrate the contribution of each.
4. We perform a thorough experimental study, and show that *GraphVid* greatly outperforms previous methods in terms of efficiency - first and foremost, the paper utilizes GNNs for efficient video understanding. We show that it successfully reduces computations while preserving much of the performance of state-of-the-art approaches that utilize computationally demanding models.

2   Related Work

2.1   Deep Learning for Video Understanding

CNNs have found numerous applications in video processing [32,49,59]. These include LSTM-based networks that perform per-frame encoding [44,50,59] and the extension of 2D convolutions to the temporal dimension, *e.g.*, 3D CNNs such as C3D [48], R2D [43] and R(2+1)D [49].

The success of the Transformer model [51] has led to the development of attention-based models for vision tasks, via self-attention modules that were used to model spatial dependencies in images. NLNet [53] was the first to employ self-attention in a CNN. With this novel attention mechanism, NLNet is possible to model long-range dependencies between pixels. The next model to be developed was GCNet [7], which simplified the NL-module, thanks to its need for fewer parameters and computations, while preserving its performance. A more prominent transition from CNNs to Transformers began with Vision Transformer (ViT) [13], which prompted research aimed at improving its effectiveness on small datasets, such as Deit [47]. Later, vision-transformers were adapted for video tasks [34,4,6,15,30,31], now crowned as the current state-of-the-art that top the leader-boards of this field.

The usage of graph representation in video understanding sparsely took place in the work of Wang [54]. They used pre-trained Resnet variants [22] for generating object bounding boxes of interest on each frame. These bounding boxes are later used for the construction of a spatio-temporal graph that describes how objects change through

time, and perform classification on top of the spatio-temporal graph with graph convolutional neural networks [26]. However, we note that the usage of a large backbone for generating object bounding boxes is harmful for performance. We intend to alleviate this by proposing a lighter graph representation. In combination of a dedicated GNN architecture, our representation greatly outperforms [54] in all metrics.

2.2   Superpixel Representation of Visual Data

Superpixels are groups of perceptually similar pixels that can be used to create visually meaningful entities while heavily reducing the number of primitives for subsequent processing steps [45]. The efficiency of the obtained representation has led to the development of many superpixel-generation algorithms for images [45]. This approach was adapted for volumetric data via the construction of supervoxels [36], which are the trivial extension to depth. These methods were adjusted for use in videos [57] by treating the temporal dimension as depth. However, this results in degraded performance, as inherent assumptions regarding neighboring points in the 3D space do not apply to videos with non-negligible motion. Recent approaches especially designed to deal with videos consider the temporal dimensions for generating superpixels that are coherent in time. Xu *et al.* [58] proposed a hierarchical graph-based segmentation method. This was followed by the work of Chang *et al.* [8], who suggested that Temporal Superpixels (TSPs) can serve as a representation of videos using temporal superpixels by modeling the flow between frames with a bilateral Gaussian process.

2.3   Graph Convolutional Neural Networks

Introduced in [26], Graph Convolutional Networks (GCNs) have been widely adopted for graph-related tasks [60,28]. The basic GCN uses aggregators, such as average and summation, to obtain a node representation given its neighbors. This basic form was rapidly extended to more complex architectures with more sophisticated aggregators. For instance, Graph Attention Networks [52] use dot-product-based attention to calculate weights for edges. Relational GCNs [41] add to this framework by also considering multiple edge types, namely, relations (such as temporal and spatial relations), and the aggregating information from each relation via separate weights in a single layer. Recently, GCNs have been adopted for tasks involving audio [12,61] and images [33,11,5]. Following the success of graph models to efficiently perform image-based tasks, we are eager to demonstrate our extension of the image-graph representation to videos.

3   *GraphVid* - A Video-Graph Representation

In this section, we introduce the methodology of *GraphVid*. First, we present our method for video-graph representation generation, depicted in Figure 2 and described in Algorithm 1. Then, we present our training methodology that utilizes this representation. Finally, we discuss the benefits of *GraphVid* and propose several augmentations.
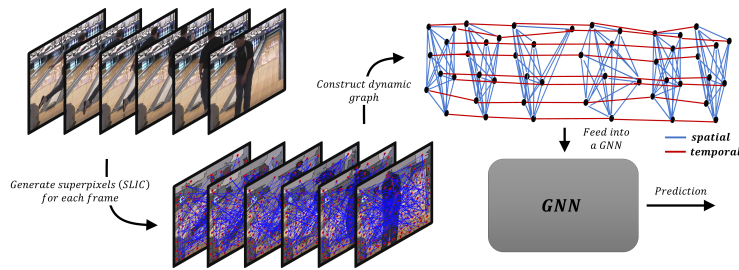
Fig. 2: The flow of *GraphVid*. Given a video clip, we generate superpixels using SLIC for each frame. The superpixels are used to construct a region-adjacency graph of a frame, with superpixels as nodes. Then, the graph sequence is connected via temporal proximities to construct a dynamic graph, which is later fed into a GNN for prediction.

3.1   Overview

In our framework, we deal with video clips that are sequences of T video frames $v \in \mathbb{R}^{T \times C \times H \times W}$. The goal is to transform v into a graph that is sufficiently informative for further processing. To achieve this, we use SLIC [2] to generate S segmented regions, called *superpixels*, over each frame. We denote each segmented region as $R_{t,i}$, where $t \in [T]$ represents the temporal frame index, and $i \in [S]$ the superpixel-segmented region index. The following is a description of how we utilize the superpixels to construct our video-graph representation.

Graph Elements - We define the undirected graph $\mathcal{G}$ as a 3-tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V} = \{R_{t,i} | t \in [T], i \in [S]\}$ is the set of nodes representing the segmented regions, $\mathcal{E}$ is the set of labeled edges (to be defined hereunder) and $\mathcal{R} = \{spatial, temporal\}$ is a set of relations as defined in [41]. Each node $R_{t,i}$ is associated with an attribute $R_{t,i}.c \in \mathbb{R}^3$ representing the mean RGB color in that segmented region. Additionally, we refer to $R_{t,i}.y$ and $R_{t,i}.x$ as the coordinates of the superpixel's centroid, which we use to compute the distances between superpixels. These distances, which will later serve as the edge attributes of the graph, are computed by

$$d_{i,j}^{t_q \to t_p} = \sqrt{\left(\frac{R_{t_q,i}.y - R_{t_p,j}.y}{H} \right)^2 + \left(\frac{R_{t_q,i}.x - R_{t_p,j}.x}{W} \right)^2}. \qquad (1)$$

Here, $t_q, t_p \in [T]$ denote frame indices, and $i, j \in [S]$ denote superpixel indices generated for the corresponding frames. The set of edges $\mathcal{E}$ is composed of: **1)** intra-frame edges (denoted $\mathcal{E}^{spatial}$) - edges between nodes corresponding to superpixels in the same frame. We refer to these as *spatial edges*. **2)** inter-frame edges (denoted $\mathcal{E}^{temporal}$) - edges between nodes corresponding to superpixels in two sequential frames. We refer to edges as *temporal edges*. Finally, the full set of edges is $\mathcal{E} = \mathcal{E}^{spatial} \cup \mathcal{E}^{temporal}$. Following is a description of how we construct both components.

Spatial Edges - In similar to [5], we generate a region-adjacency graph for each frame, with edge attributes describing the distances between superpixel centroids. The notation $\mathcal{E}_t^{spatial}$ refers to the set of the spatial-edges connecting nodes corresponding to

superpixels in the frame t, and $\mathcal{E}^{spatial} = \bigcup_{t=1}^{T} \mathcal{E}_t^{spatial}$. Each edge $e_{i,j}^t \in \mathcal{E}^{spatial}$ is associated with an attribute that describes the euclidean distance between the two superpixel centroids i and j in frame t, that is, $d_{i,j}^{t \to t}$. These distances provide information about the relations between the superpixels. Additionally, the distances are invariant to rotations and image-flips, which eliminates the need for those augmentations. Note that normalization of the superpixels' centroid coordinates is required in order to obscure information regarding the resolution of frames, which is irrelevant for many tasks, such as action classification. In Figure 3, we demonstrate the procedure of spatial edge generation for a cropped image that results in a partial graph of the whole image. Each superpixel is associated with a node, which is connected via edges to other adjacent nodes (with the distances between the superpixels' centroids serving as edge attributes).
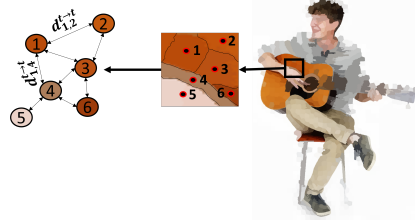

Fig. 3: Spatial edge generation. First, superpixels are generated. Each superpixel is represented as a node, which is connected via its edges to other such nodes within a frame. Each node is assigned the mean color of the respective segmented region, and each edge is assigned the distances between the superpixel centroids connected by that edge.

Temporal Edges - In modeling the temporal relations, we aim to connect nodes that tend to describe the same objects in subsequent frames. To do so, we rely on the assumption that in subsequent frames, such superpixels are attributed similar colors and the same spatial proximity. To achieve this, for each superpixel $R_{t,i}$, we construct a neighborhood $\mathcal{N}_{t,i}$ that contains superpixels from its subsequent frame whose centroids have a proximity of at most $d_{proximity} \in (0,1]$ with respect to the euclidean distance. Then, we find the superpixel with the most similar color in this neighborhood. As a result, the tth frame is associated with the set of edges $\mathcal{E}_{t \to t+1}^{temporal}$ that model temporal relations with its subsequent frame, formally:

$$\mathcal{N}_{t,i} = \{R_{t+1,j} | d_{i,j}^{t \to t+1} < d_{proximity}\}, \tag{2}$$

$$neighbor(R_{t,i}) = \underset{R_{t+1,j} \in \mathcal{N}_{t,i}}{argmin} |R_{t,i}.c - R_{t+1,j}.c|_2, \tag{3}$$

$$\mathcal{E}_{t \to t+1}^{temporal} = \{(R_{t,i}, temporal, neighbor(R_{t,i}) | i \in [S]\}. \tag{4}$$

Equipped with these definitions, we define the set of temporal edges connecting nodes corresponding to superpixels in frame t to superpixels in frame $t+1$ as the union of the temporal edge sets generated for all the frames: $\mathcal{E}^{temporal} = \bigcup_{t=1}^{T-1} \mathcal{E}_{t \to t+1}^{temporal}$.

Algorithm 1 Graph Generation

Input: $v \in \mathbb{R}^{T \times C \times H \times W}$ ▷ The input video clip
Parameters: $S \in \mathbb{N}$ ▷ Number of superpixels per frame
            $d_{proximity} \in (0, 1]$ ▷ Diameter of neighborhoods
Output: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ ▷ A video-graph
$\mathcal{V}, \mathcal{V}_{last}, \mathcal{E}^{spatial}, \mathcal{E}^{temporal} \leftarrow \emptyset, \emptyset, \emptyset, \emptyset$
for $t \in [T]$ **do**
     $SP \leftarrow SLIC(v[t], S)$
     $\mathcal{V} \leftarrow \mathcal{V} \cup SP$
     $\mathcal{E}^{spatial} \leftarrow \mathcal{E}^{spatial} \cup regionAdjacetEdges(SP)$
     $\mathcal{E}^{temporal}_{t-1 \rightarrow t} \leftarrow \emptyset$
     for $R_{t-1,i} \in \mathcal{V}_{last}$ **do**
         $\mathcal{N}_{t-1,i} \leftarrow \{R_{t,j} | d^{t-1 \rightarrow t}_{i,j} < d_{proximity}\}$
         $nn_{t-1,i} \leftarrow \text{argmin}_{R_{t,j} \in \mathcal{N}_{t,i}} |R_{t,i}.c - R_{t,j}.c|_2)$
         $\mathcal{E}^{temporal}_{t-1 \rightarrow t} \leftarrow \mathcal{E}^{temporal}_{t-1 \rightarrow t} \cup \{(R_{t-1,i}, temporal, nn_{t-1,i})\}$
     end for
     $\mathcal{E}^{temporal} \leftarrow \mathcal{E}^{temporal} \cup \mathcal{E}^{temporal}_{t-1 \rightarrow t}$
     $\mathcal{V}_{last} \leftarrow SP$
end for
return $\mathcal{G} = (\mathcal{V}, \mathcal{E} = \mathcal{E}^{spatial} \cup \mathcal{E}^{temporal}, \mathcal{R} = \{spatial, tempo\})$

3.2 Model Architecture

In order to model both the spatial and temporal relations between superpixels, our model primarily relies on the Neural Relational Model [41], which is an extension of GCNs [26] to large-scale relational data. In a Neural Relational Model, the propagation model for calculating the forward-pass update of a node, denoted by v_i, is defined as

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right),$$
(5)

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i under relation $r \in \mathcal{R}$ (not to be confused with the notation $\mathcal{N}_{t,i}$ from Eq. 2). $c_{i,r}$ is a problem-specific normalization constant that can either be learned or chosen in advance (such as $c_{i,r} = |\mathcal{N}_i^r|$). To incorporate edge features, we adapt the approach proposed in [10], that concatenates node and edge attributes as a layer's input, yielding the following:

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} W_r^{(l)} [h_j^{(l)}, e_{i,j}] + W_0^{(l)} h_i^{(l)} \right),$$
(6)

where $e_{i,j}$ is the feature of the edge connecting nodes v_i, v_j.

3.3 Augmentations

We introduce a few possible augmentations that we found useful for training our model as they improved the generalization.

Additive Gaussian Edge Noise (AGEN) - Edge attributes represent distances between superpixel centroids. The coordinates of those centroids may vary due to different superpixel shapes with different centers of mass. To compensate for this, we add a certain amount of noise to each edge attribute. Given a hyper-parameter σ_{edge}, for each edge attribute $e_{u,v}$ and for each training iteration, we sample a normally distributed variable $z_{u,v} \sim N(0, \sigma_{edge})$ that is added to the edge attribute.

Additive Gaussian Node Noise (AGNN) - Node attributes represent the colors of regions in each frame. Similar to edge attributes, the mean color of each segmented region may vary due to different superpixel shapes. To compensate for this, we add a certain amount of noise to each node attribute. Given a hyper-parameter σ_{node}, for each node attribute $v.c$ of dimension d_c and for each training iteration, we sample a normally distributed variable $z_v \sim N_{d_c}(0, \sigma_{node} \cdot I_{d_c})$ that is added to the node attribute.

Random Removal of Spatial Edges (RRSE) - This augmentation tends to mimic the regularization effect introduced in DropEdge [39]. Moreover, since the removal of edges leads to fewer message-passings in a GCN, this also accelerates the training and inference. To perform this, we choose a probability $p_{edge} \in [0, 1]$. Then, each edge e is preserved with a probability of p_{edge}.

Random Removal of Superpixels (RRS) - SLIC [2] is sensitive to its initialization. Consequently, each video clip may have several graph representations during different training iterations and inference. This can be mitigated by removing a certain amount of superpixels. The outcome is fewer nodes in the corresponding representative graph, as well as fewer edges. Similar to RRSE, we choose a probability $p_{node} \in [0, 1]$ so that each superpixel is preserved with a probability of p_{node}.

3.4   Benefits of *GraphVid*

Invariance - The absence of coordinates leads to invariance in the spatial dimension. It is evident that such a representation is invariant to rotations and flips since the relations between different parts of the image are solely characterized by distances. This, in turn, obviates the need to perform such augmentations during training.

Efficiency - We argue that our graph-based representation is more efficient than raw frames. To illustrate this, let T, C, H and W be the dimensions of a clip; that is, the number of frames, number of channels, height and width of a frame, respectively. Correspondingly, the raw representation requires $T \cdot C \cdot H \cdot W$. To calculate the size of the graph-video, let S be the number of superpixels in a frame. By construction, there are at most $4 \cdot S$ edges in each frame because SLIC constraints each to have 4 neighbors. Each edge contains 3 values, corresponding to the distance on the grid, source and target nodes. Additionally, there are, at most, S edges between every temporal step. This results in $3 \cdot (4 \cdot S + (T - 1) \cdot S) + C \cdot T \cdot S$ parameters in total. Typically, the second requires much fewer parameters because we choose S so that $S \ll H \cdot W$.

Prior Knowledge Incorporation - Optical-flow and over-segmentation are encoded within the graph-video representation using the inter-frame and intra-frame edges. This incorporates strong prior knowledge within the resultant representation. For example, optical-flow dramatically improved the accuracy in the two-stream methodology that was proposed in [43]. Additionally, over-segmentation using superpixels has been found useful as input features for machine learning models due to the limited loss of important details, accompanied by a dramatic reduction in the expended time by means of reducing the number of elements of the input [21,11,5].

4 Experiments

We validated *GraphVid* on 2 human-action-classification benchmarks. The goal of human action classification is to determine the human-involved action that occurs within a video. The objectives of this empirical study were twofold:

– Analyze the impact of the various parameters on the accuracy of the model.
– As we first and foremost target efficiency, we sought to examine the resources' consumption of *GraphVid* in terms of Floating Point Operations (FLOPs). We followed the conventional protocol [16], which uses single-clip FLOPs as a basic unit of computational cost. We show that we are able to achieve a significant improvement in efficiency over previous methods while preserving state-of-the-art performance.

4.1 Setup

Datasets - We use two common datasets for action classification: *Kinetics-400 (K400)* [23] and *Charades* [42]. Kinetics-400 [23] is a large-scale video dataset released in 2017 that contains 400 classes, with each category consisting of more than 400 videos. It originally had, in total, around 240K, 19K, and 38K videos for training, validation and testing subsets, respectively. Kinetics is gradually shrinking over time due to videos being taken offline, making it difficult to compare against less recent works. We used a dataset containing 208K, 17K and 33K videos for training, validation and test respectively. We report on the most recently available videos. Each video lasts approximately 10 seconds. The Charades dataset [42] is composed of 9,848 videos of daily indoor activities, each of an average length of 30 seconds. In total, the dataset contains 66,500 temporal annotations for 157 action classes. In the standard split, there are 7,986 training videos and 1,863 validation videos, sampled at 12 frames per second. We follow prior arts by reporting the Top-1 and Top-5 recognition accuracy for Kinetics-400 and mean average precision (mAP) for Charades.

Network Architecture and Training - We use GNN variants and feed each of them with our video-graphs. Specifically, we consider Graph Convolutional Networks [26] (GCNs), Graph Attention Networks [52] (GATs) and Relational Graph Convolutional Networks [41] (RGCNs). The general architecture of our backbones is depicted in Fig. 4. It consists of 2 fully-connected (FC) layers with exponential linear unit (ELU) activations that project the node features into a $256D$ feature space. Then come 4 layers of
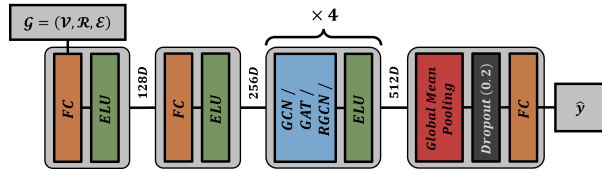
Fig. 4: The general graph neural network architecture we use in our experiments.

the corresponding GNN layer (either GCN, GAT or RGCN along with an edge feature concatenation from Eq. 6) with a hidden size of 512 with ELU activations, followed by global mean pooling, dropout with a probability of 0.2 and a linear layer whose output is the predicted logits. For the GAT layers, we use 4 attention heads in each layer, and average the attention heads' results to obtain the desired hidden layer size. For the RGCN layers, we specify 2 relations, which correspond to the spatial and temporal relations, as described in Section 3. We use the Adam [25] with a constant learning rate of $1e - 3$ for optimization. While choosing this architecture, the core idea was to keep the architecture simple and shallow, while changing the interaction module to better model the relations between parts of the clip. We divide the videos into clips using a sliding window of 20 frames, using a stride of 2 between consecutive frames and a stride of 10 between clips. In all the experiments, we used a fixed batch size of 200.

Inference - At the test phase, we use the same sliding window methodology as in the training. We follow the common practice of processing multiple views of a long video and average per-view logits to obtain the final results. The views are drawn uniformly across the temporal dimension of the video, without spatial cropping. The number of views is determined by the validation dataset.

Implementation Details - All experiments were run on a Ubuntu 18.04 machine with Intel i9-10920X, 93GB RAM and 2 GeForce RTX 3090 GPUs. Our implementation of *GraphVid* is in Python3. To generate superpixels, we use *fast-slic* [24] with the AVX2 instruction set. To train the graph neural models, we use Pytorch-Geometric [19]. We use a fixed seed for SLIC and cache the generated graphs during the first training epochs in order to further reduce the computations. We also store the edge indexes as int16 instead of int64 in order to reduce the memory footprint. Eventually, the memory footprints of the cached datasets is comparable to those of the original ones.

4.2   Ablation Study

We conduct an in-depth study on Kinetics-400 to analyze the performance gain contributed by incorporating the different components of *GraphVid*.

Graph Neural Network Variants and Number of Superpixels per Frame - We assess the performance of different GNN variants: GCN [26] is trained without edge relations (*i.e.* temporal and spatial edges are treated via the same weights). GAT [52] is trained by employing the attention mechanism for neighborhood aggregation without edge relations. RGCN [41] is trained with edge relations, as described in Section 3.2.

The results of the action classification on K-400 are shown in Figure 5. In this series, the number of views is fixed at 8, which is the number of views that was found to be most effective for the validation set. For all variants, increasing the number of superpixels per frame (S) contributes to the accuracy. We notice a significant improvement in accuracy for the lower range of the number of superpixels, while the accuracy begins to saturate for $S \geq 650$. Increasing further the number of superpixels leads to bigger inputs, which require more computations. As our goal is to maximize the efficiency, we do not experiment with larger inputs in this section. We further present in Table 1 the
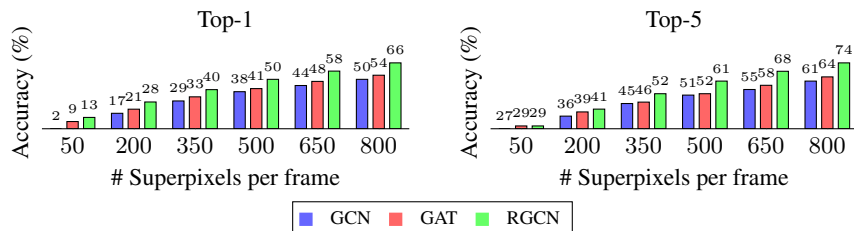

Fig. 5: The effect of varying the number of superpixels on test accuracy on K-400.

models' specifications for 800 superpixels, which is the best-performing configuration in this series of experiments. Unsurprisingly, the GCN variant requires the least amount of computations. Meanwhile, the RGCN variant requires fewer computations than GAT and achieves a higher level of accuracy. We conclude that it is beneficial to incorporate edge relations when wishing to encode temporal and spatial relations in videos, and that those features are not easily learned by heavy computational models, such as GAT.

Table 1: Comparison of model specifications for various architectures. We report the Top-1 and Top-5 accuracy on Kinetics-400.

Model	Top-1	Top-5	FLOPs ($\cdot 10^9$)	Params ($\cdot 10^6$)
GCN	50.1	61.6	28	2.08
GAT	54.7	64.5	56	3.93
$RGCN$	66.2	74.1	42	2.99

Augmentations - We assessed the impact of augmentations on the performance and their ability to alleviate over-fitting. For this purpose, we chose the best configuration obtained from the previous experiments, that is, RGCN with 800 superpixels per frame, and trained it while adding one augmentation at a time. The results of this series are depicted in Figure 7. Each graph shows the level of accuracy reached by training the model with one of the parameters that control the augmentation.

We begin with the analysis of the AGEN and AGNN, both relate to the addition of Gaussian noise to the graph components, with the corresponding parameters con-
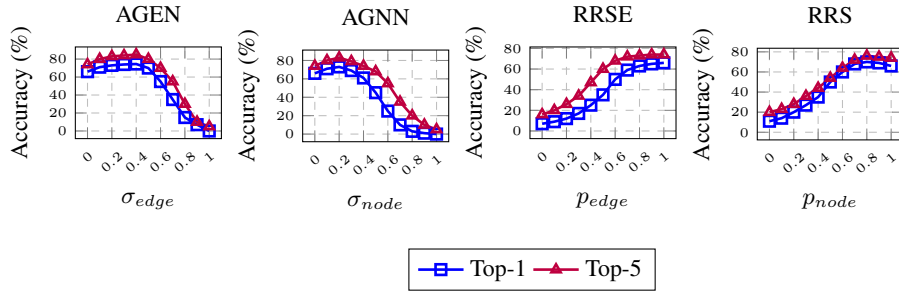
Fig. 6: The impact of the proposed augmentations on test accuracy of Kinetics-400: Additive Gaussian edge noise (AGEN). Additive Gaussian node noise (AGNN). Random removal of spatial edges (RRSE). Random removal of superpixels (RRS).

trolling the standard deviations. Their impact is unnoticeable as these parameters head towards 0, since lower values reflect the scenarios in which little or no augmentations are applied. Slightly increasing the parameter brings about a gradual improvement in the accuracy, until a turning point is reached, after which the level of accuracy declines until it reaches $\sim \frac{1}{400}$, which resembles a random classifier. The decrease in accuracy stems from the noise obscuring the original signal, allegedly forcing the classifier to classify ungeneralizable noise. For RRSE and RRS, the random removal of spatial edges harms the accuracy of the model. This finding leads us to conclude that spatial edges encode meaningful information about relations between the entities. Moreover, slightly removing the nodes contributes to the level of accuracy, reaching a peak at $p_{node} \approx 0.8$. To conclude, we present the values that lead to the best Top-1 accuracy score in Table 2.

Table 2: Augmentation parameters and their optimized values.

Param	σ_{edge}	σ_{node}	p_{edge}	p_{node}
Value	0.4	0.2	1	0.8
Top-1	74.5	73	66	70
Top-5	85	83	74	76

4.3   Comparison to the State-of-the-Art

Kinetics-400 - We present the K-400 results for our RGCN variant in Table 3 and Figure 7a, along with comparisons to previous arts, including convolutional-based and transformer-based methods. Our results are denoted RGCN-d, where d represents the number of superpixels. Additionally, we use the set of augmentations with the parameters from Table 2. First, when the RGCN-800 model is trained with the full set of augmentations (denoted Full-Aug), it achieves a significantly higher Top-1 accuracy
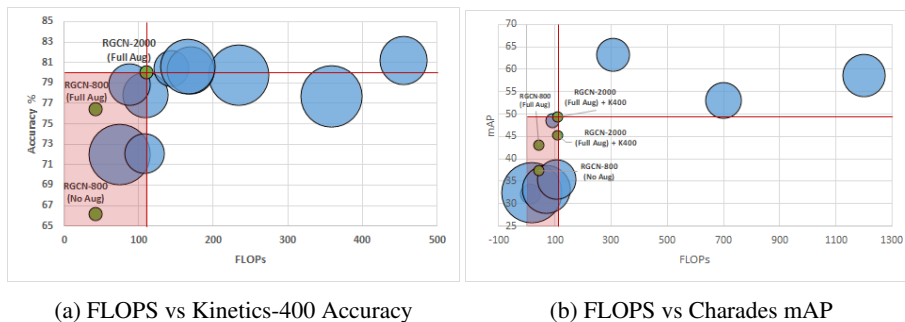
(a) FLOPS vs Kinetics-400 Accuracy   (b) FLOPS vs Charades mAP

Fig. 7: **Model FLOPs vs. performance -** Green bubbles indicates *GraphVid* variants, radius indicates the number of parameters. To avoid overloading, identities of the other models are omitted. For both datasets, RGCN-2000 with the full set of augmentations is on par with the state-of-the-art, while greatly reducing model size and FLOPs.

than when it is trained without any augmentation (denoted No-Aug) or when each augmentation is applied individually. These results demonstrate the effectiveness of our model and that our augmentations can alleviate overfitting and improve the generalization over the test set. Second, all our RGCNs require orders-of-magnitude fewer computations than the previous arts, as well as more than $\times 10$ fewer parameters.

Table 3: Comparisons to state-of-the-art on the K-400 dataset. We report the Top-1 and Top-5 accuracies. The top section of the table depicts convolution-based models, The middle section depicts transformer-based models, and the bottom section depicts ours.

Method	Top-1	Top-5	Views	FLOPs $(\cdot 10^9)$	Param $(\cdot 10^6)$
SlowFast R101+N [17]	79.8	93.9	30	234	59.9
X3D-XXL R101+N [16]	80.4	94.6	30	144	20.3
MViT-B, 32×3 [15]	80.2	94.4	5	170	36.6
TimeSformer-L [6]	80.7	94.7	3	2380	121.4
ViT-B-VTN [34]	78.6	93.7	1	4218	11.04
ViViT-L/16x2 [4]	80.6	94.7	12	1446	310.8
Swin-S [31]	80.6	94.5	12	166	49.8
RGCN-800 (No / Full Aug)	66.2 / 76.4	74.1 / 91.1	8	**42**	**2.57**
RGCN-2000 (Full Aug)	80.0	94.3	8	**110**	**2.57**

Charades - We train RGCN variants with 800 and 2000 superpixels with the set of augmentations found in Table 2. We also follow prior arts [17,15] by pre-training on K-400 followed by replacing the last FC layer and fine-tuning on Charades. Table 4 and Figure 7b show that when our RGCN model is trained with 2000 superpixels, its mAP score is comparable to the current state-of-the-art, but this score is reached with orders-of-magnitude fewer computations and using considerably fewer parameters.

Table 4: Comparisons to state-of-the-art on the Charades multi-label dataset. We report the mAP scores as more than one ground truth action is possible.

Method	mAP	FLOPs ($\cdot 10^9$)	Params ($\cdot 10^6$)
MoVieNet-A2 [27]	32.5	6.59	4.8
MoVieNet-A4 [27]	48.5	90.4	4.9
TVN-1 [38]	32.2	13	11.1
TVN-4 [38]	35.4	106	44.2
AssembleNet-50 [40]	53.0	700	37.3
AssembleNet-101 [40]	58.6	1200	53.3
SlowFast 16×8 R101 [17]	45.2	7020	59.9
RGCN-800 (No Aug/Full Aug)	37.4 / 43.1	**42**	**2.57**
RGCN-2000 (Full Aug/+K400)	45.3 / 49.4	**110**	**2.57**

4.4   Video-Graph Generation Run-Time

The transition into a video-graph representation requires the consideration of the time needed for generating it. In Figure 8, we measured the average time needed using our setup, which include the whole pipeline: **1.** Superpixels calculation, and **2.** Graph structure generation, that is, creating edges between adjacent super-pixels and features calculation as described in Section 3. Interestingly, the first step is relatively short compared to the second. Apparently, the optimized *fast-slic* [24] performs well, while the search for adjacent superpixels is time consuming. This opens the possibilities of further optimization.
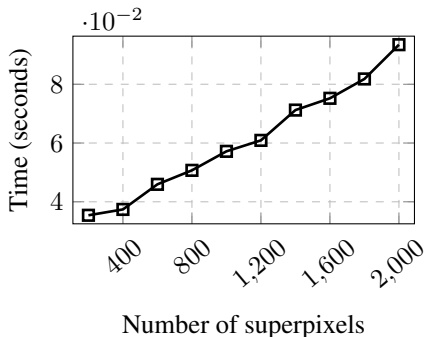

Fig. 8: Time of generation depending on the number of superpixels.

5   Conclusions and Future Work

In this paper, we present *GraphVid*, a graph video representations that enable video-processing via graph neural networks. Furthermore, we propose a relational graph convolutional model that suits this representation. Our experimental study demonstrates this model's efficiency in performing video-related tasks while achieving comparable performance to the current state-of-the-art. An interesting avenue for future work is to explore new graph representations of videos, including learnable methods. Additionally, we consider the development of new dedicated graph neural models for processing the unique and dynamic structure of the video-graph as an interesting research direction. Finally, unified models for image and video understanding that disregard temporal edges could be explored in order to take advantage of the amount of data in both worlds.

References

1. Abadal, S., Jain, A., Guirado, R., López-Alonso, J., Alarcón, E.: Computing graph neural networks: A survey from algorithms to accelerators. ACM Computing Surveys (CSUR) **54**(9), 1–38 (2021)
2. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels. Tech. rep. (2010)
3. Akbari, H., Yuan, L., Qian, R., Chuang, W.H., Chang, S.F., Cui, Y., Gong, B.: Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. arXiv preprint arXiv:2104.11178 (2021)
4. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: Vivit: A video vision transformer. arXiv preprint arXiv:2103.15691 (2021)
5. Avelar, P.H., Tavares, A.R., da Silveira, T.L., Jung, C.R., Lamb, L.C.: Superpixel image classification with graph attention networks. In: SIBGRAPI. pp. 203–209. IEEE (2020)
6. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding? arXiv preprint arXiv:2102.05095 (2021)
7. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In: ICCV Workshops. pp. 0–0 (2019)
8. Chang, J., Wei, D., Fisher, J.W.: A video representation using temporal superpixels. In: CVPR. pp. 2051–2058 (2013)
9. Chen, Y., Kalantidis, Y., Li, J., Yan, S., Feng, J.: Multi-fiber networks for video recognition. In: ECCV. pp. 352–367 (2018)
10. Corso, G., Cavalleri, L., Beaini, D., Liò, P., Veličković, P.: Principal neighbourhood aggregation for graph nets. arXiv preprint arXiv:2004.05718 (2020)
11. Dadsetan, S., Pichler, D., Wilson, D., Hovakimyan, N., Hobbs, J.: Superpixels and graph convolutional neural networks for efficient detection of nutrient deficiency stress from aerial imagery. In: CVPR. pp. 2950–2959 (2021)
12. Dokania, S., Singh, V.: Graph representation learning for audio & music genre classification. arXiv preprint arXiv:1910.11117 (2019)
13. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
14. Du Tran, H.W., Torresani, L., Ray, J., Lecun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition.(2017) (2017)
15. Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., Feichtenhofer, C.: Multiscale vision transformers. arXiv preprint arXiv:2104.11227 (2021)
16. Feichtenhofer, C.: X3d: Expanding architectures for efficient video recognition. In: CVPR. pp. 203–213 (2020)
17. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: ICCV. pp. 6202–6211 (2019)
18. Fernández, D., Varas, D., Espadaler, J., Masuda, I., Ferreira, J., Woodward, A., Rodríguez, D., Giró-i Nieto, X., Carlos Riveiro, J., Bou, E.: Vits: video tagging system from massive web multimedia collections. In: ICCV Workshops. pp. 337–346 (2017)
19. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
20. Girdhar, R., Carreira, J., Doersch, C., Zisserman, A.: Video action transformer network. In: CVPR. pp. 244–253 (2019)
21. Gonzalo-Martin, C., Garcia-Pedrero, A., Lillo-Saavedra, M., Menasalvas, E.: Deep learning for superpixel-based classification of remote sensing images (September 2016), `http://proceedings.utwente.nl/401/`

22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
23. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017)
24. Kim, A.: fast-slic. https://github.com/Algy/fast-slic (2019)
25. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
26. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
27. Kondratyuk, D., Yuan, L., Li, Y., Zhang, L., Tan, M., Brown, M., Gong, B.: Movinets: Mobile video networks for efficient video recognition. In: CVPR. pp. 16020–16030 (2021)
28. Kumar, A., Singh, S.S., Singh, K., Biswas, B.: Link prediction techniques, applications, and performance: A survey. Physica A: Statistical Mechanics and its Applications **553**, 124289 (2020)
29. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
30. Li, X., Zhang, Y., Liu, C., Shuai, B., Zhu, Y., Brattoli, B., Chen, H., Marsic, I., Tighe, J.: Vidtr: Video transformer without convolutions. arXiv preprint arXiv:2104.11746 (2021)
31. Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., Hu, H.: Video swin transformer. arXiv preprint arXiv:2106.13230 (2021)
32. Mittal, S., et al.: A survey of accelerator architectures for 3d convolution neural networks. Journal of Systems Architecture p. 102041 (2021)
33. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: CVPR. pp. 5115–5124 (2017)
34. Neimark, D., Bar, O., Zohar, M., Asselmann, D.: Video transformer network. arXiv preprint arXiv:2102.00719 (2021)
35. Oprea, S., Martinez-Gonzalez, P., Garcia-Garcia, A., Castro-Vargas, J.A., Orts-Escolano, S., Garcia-Rodriguez, J., Argyros, A.: A review on deep learning techniques for video prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
36. Papon, J., Abramov, A., Schoeler, M., Worgotter, F.: Voxel cloud connectivity segmentation-supervoxels for point clouds. In: CVPR. pp. 2027–2034 (2013)
37. Pareek, P., Thakkar, A.: A survey on video-based human action recognition: recent updates, datasets, challenges, and applications. Artificial Intelligence Review **54**(3), 2259–2322 (2021)
38. Piergiovanni, A., Angelova, A., Ryoo, M.S.: Tiny video networks. Applied AI Letters p. e38 (2019)
39. Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. arXiv preprint arXiv:1907.10903 (2019)
40. Ryoo, M.S., Piergiovanni, A., Tan, M., Angelova, A.: Assemblenet: Searching for multi-stream neural connectivity in video architectures. arXiv preprint arXiv:1905.13209 (2019)
41. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: ESWC. pp. 593–607. Springer (2018)
42. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: Crowdsourcing data collection for activity understanding. In: ECCV. pp. 510–526. Springer (2016)
43. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. arXiv preprint arXiv:1406.2199 (2014)
44. Srivastava, N., Mansimov, E., Salakhudinov, R.: Unsupervised learning of video representations using lstms. In: ICML. pp. 843–852. PMLR (2015)

45. Stutz, D., Hermans, A., Leibe, B.: Superpixels: An evaluation of the state-of-the-art. Computer Vision and Image Understanding **166**, 1–27 (2018)
46. Suarez, J.J.P., Naval Jr, P.C.: A survey on deep learning techniques for video anomaly detection. arXiv preprint arXiv:2009.14146 (2020)
47. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML. pp. 10347–10357. PMLR (2021)
48. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: ICCV. pp. 4489–4497 (2015)
49. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: CVPR. pp. 6450–6459 (2018)
50. Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., Baik, S.W.: Action recognition in video sequences using deep bi-directional lstm with cnn features. IEEE access **6**, 1155–1166 (2017)
51. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
52. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
53. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR. pp. 7794–7803 (2018)
54. Wang, X., Gupta, A.: Videos as space-time region graphs. In: Proceedings of the European conference on computer vision (ECCV). pp. 399–417 (2018)
55. Xie, R., Liu, Z., Jia, J., Luan, H., Sun, M.: Representation learning of knowledge graphs with entity descriptions. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 30 (2016)
56. Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K.: Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: ECCV. pp. 305–321 (2018)
57. Xu, C., Corso, J.J.: Evaluation of super-voxel methods for early video processing. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1202–1209 (2012). https://doi.org/10.1109/CVPR.2012.6247802
58. Xu, C., Xiong, C., Corso, J.J.: Streaming hierarchical video segmentation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) Computer Vision – ECCV 2012. pp. 626–639. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
59. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR. pp. 4694–4702 (2015)
60. Zhang, D., Yin, J., Zhu, X., Zhang, C.: Network representation learning: A survey. IEEE transactions on Big Data **6**(1), 3–28 (2018)
61. Zhang, S., Qin, Y., Sun, K., Lin, Y.: Few-shot audio classification with attentional graph neural networks. In: Interspeech. pp. 3649–3653 (2019)