Computer Vision and Image Understanding 206 (2021) 103187

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Video action detection by learning graph-based spatio-temporal interactions

Matteo Tomei [a,*], Lorenzo Baraldi [a], Simone Calderara [a], Simone Bronzin [b], Rita Cucchiara [a]

[a] University of Modena and Reggio Emilia, via Pietro Vivarelli 10, Modena 41125, Italy
[b] METALIQUID S.R.L., Via Giosue Carducci 26, Milano 20123, Italy

ARTICLE INFO

Communicated by Nikos Paragios

MSC:
68T10
68T45

Keywords:
Video understanding
Action detection
Graph learning

ABSTRACT

Action Detection is a complex task that aims to detect and classify human actions in video clips. Typically, it has been addressed by processing fine-grained features extracted from a video classification backbone. Recently, thanks to the robustness of object and people detectors, a deeper focus has been added on relationship modeling. Following this line, we propose a graph-based framework to learn high-level interactions between people and objects, in both space and time. In our formulation, spatio-temporal relationships are learned through self-attention on a multi-layer graph structure which can connect entities from consecutive clips, thus considering long-range spatial and temporal dependencies. The proposed module is backbone independent by design and does not require end-to-end training. Extensive experiments are conducted on the AVA dataset, where our model demonstrates state-of-the-art results and consistent improvements over baselines built with different backbones. Code is publicly available at https://github.com/aimagelab/STAGE_action_detection.

1. Introduction

Understanding people actions in video clips is an open problem in computer vision, which has been addressed for more than twenty years (Bobick and Davis, 2001; Herath et al., 2017). In the past, this task was tackled through handcrafted features designed for specific actions (Laptev, 2005; Vezzani et al., 2009). Recently, the video action detection task (Sun et al., 2018; Ulutan et al., 2020; Yang et al., 2019) was introduced along with deep architectures able to extract fine-grained and discriminative spatio-temporal features, to represent video chunks in a compact and manageable form. This has motivated recent efforts to design novel backbones for video feature extraction (Feichtenhofer et al., 2019; Tran et al., 2018; Wu et al., 2019). On the other hand, higher-level reasoning is necessary for detecting and understanding human actions.

Interestingly, the performances of video action detection networks that take inspiration from object detection architectures are still far from being satisfactory. For example, it would be difficult to recognize whether a person is *watching* someone just by looking at a bounding box around him, without considering the context. This can be partly explained by the lack of proper context understanding of the previous works, as they cannot model the relationships between actors and surrounding elements (Ulutan et al., 2020). Also, the presence of objects and other people in the scene, together with their behaviors, influences the understanding of the actor at hand.

High-level reasoning is necessary not only at the spatial level, to model relations between close entities, but also in time: most of the existing backbones can handle small temporal variations, without modeling long-term temporal relationships.

Following these premises, we devise a high-level module for video action detection which considers interactions between different people in the scene and interactions between actors and objects. Further, it can also take into account temporal dependencies by connecting consecutive clips during learning and inference. The same module can be stacked multiple times to form a multi-layer structure (Fig. 1). In this manner, the overall temporal receptive field can be arbitrarily increased to model long-range dependencies. Since our method works at the feature level, it can expand its temporal receptive field without dramatically increasing its computational requirements. Our solution can exploit existing backbones for feature extraction and can achieve state-of-the-art results without an end-to-end finetuning of the underlying backbone.

Previous works in action analysis have already tried to exploit graph-based representations (Wang and Gupta, 2018; Zhang et al., 2019), to model relationships with the context (Girdhar et al., 2019; Ulutan et al., 2020) and to exploit long-term temporal relations (Wu et al., 2019): our proposal merges all these insights in a single module, which is independent of the feature extraction layers and works on pre-computed representations. Moreover, our model is the first to employ a learning-based approach also on graph edges. We test our model on the Atomic Visual Actions (AVA) dataset (Gu et al., 2018), which represents a challenging test-bed for recognizing human actions and exploiting the role of context, and provide experiments on J-HMDB-21

* Corresponding author.
E-mail address: matteo.tomei@unimore.it (M. Tomei).

https://doi.org/10.1016/j.cviu.2021.103187
Received 29 June 2020; Received in revised form 1 February 2021; Accepted 24 February 2021
Available online 27 February 2021

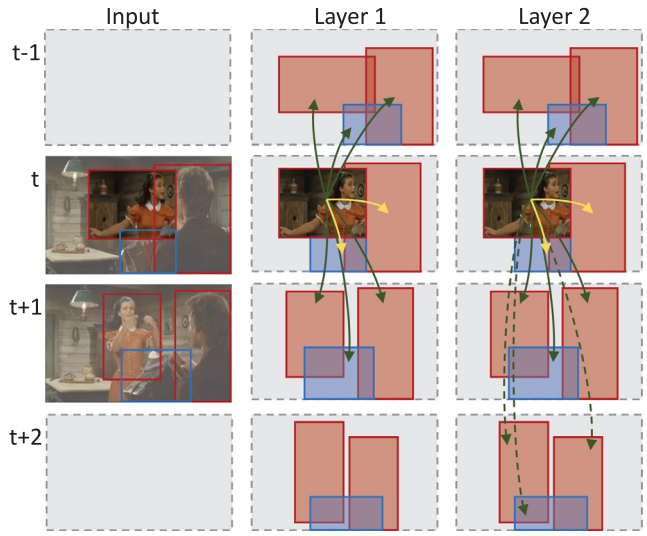

Fig. 1. We propose a graph-based module for video action detection which encodes relationships between actors and objects in a spatio-temporal neighborhood. Multiple layers of the module generate indirect edges between temporally distant entities, increasing the temporal receptive field.

(Jhuang et al., 2013) and UCF101-24 (Soomro et al., 2012). We demonstrate that our approach increases the performance of three different video backbones, reaching state-of-the-art results on AVA 2.1 and AVA 2.2.

Contributions. To sum up, our contributions are as follows:

- We propose a novel module for video action detection, which considers spatio-temporal relationships between actors and objects.
- The proposed module is based on a spatio-temporal graph representation of the video, which is learned through self-attention operations. Further, multiple instances of the proposed module can be stacked together to obtain a greater temporal receptive field. The overall model is independent of the feature extraction stage and does not need end-to-end training to achieve state-of-the-art results.
- Extensive experiments on the challenging AVA dataset (Gu et al., 2018) validate our approach and its components, demonstrating better performance with respect to other end-to-end models. In particular, our proposal achieves 29.8 and 31.8 mAP on AVA 2.1 and AVA 2.2, respectively, surpassing all previous approaches.

2. Related work

Deep networks for video understanding. CNNs are currently the state-of-the-art approach to extract spatio-temporal features for video processing and understanding (Carreira and Zisserman, 2017; Tran et al., 2015; Varol et al., 2017; Xie et al., 2018). Most of the approaches for spatio-temporal feature extraction have employed either full 3D convolutional kernels, or a combination of 2D spatial kernels and 1D temporal filters (Feichtenhofer et al., 2016a; Qiu et al., 2017; Tran et al., 2018). Despite convolutions are capable of extracting temporal features to some extent, it is still a common practice to integrate both RGB and optical flow inputs in two separate streams (Feichtenhofer et al., 2016b; Simonyan and Zisserman, 2014; Wang et al., 2016) to capture appearance and motion respectively. Recently, some works have proposed architectural variations to improve the feature extraction capabilities of the network. Li et al. (2019) have proposed to employ 2D convolutional kernels which slide over the three 2D projections of a spatio-temporal tensor; (Hussein et al., 2019), instead, have developed a multi-scale temporal convolution approach which uses different kernel sizes and dilation rates to better capture temporal dependencies. On the same line, Feichtenhofer et al. (2019) has proposed a two-pathway network, with a low frame rate path that focuses on the extraction of spatial information and a high frame rate path that encodes motion.

Detecting actions in space and time. Detecting actions is a fundamental step towards human behavior understanding in videos. Towards this goal, two problems have been analyzed in the last few years: temporal action detection and spatio-temporal action detection. The former task aims to segment the temporal interval in which the action takes place (Caba Heilbron et al., 2015; Sigurdsson et al., 2016; Xu et al., 2017). The latter, instead, is intended to detect people in space and time and to classify their actions (Gu et al., 2018; Jhuang et al., 2013; Soomro et al., 2012; Soomro and Zamir, 2014). Seminal works in action detection and recognition have already investigated the role of context and that of modeling the interactions with objects (Gupta and Davis, 2007; Gupta et al., 2009; Prest et al., 2012; Escorcia and Niebles, 2013) to improve recognition. Other approaches proposed to split the action localization task into spatial and temporal search (Kläser et al., 2010).

Recent approaches have tackled the spatio-temporal action detection task by exploiting human proposals coming from pre-trained image detectors (Feichtenhofer et al., 2019; Wu et al., 2019) and replicating them in time to build straight spatio-temporal tubes; others have extended image detection architectures to infer more precise spatio-temporal tubelets (Gkioxari and Malik, 2015; Hou et al., 2017; Kalogeiton et al., 2017; Saha et al., 2017). (Gu et al., 2018) proposed a baseline exploiting an I3D network encoding RGB and flow data separately, along with a Faster R-CNN (Ren et al., 2015), to jointly learn action proposals and labels. Ulutan et al. (2020) suggested combining actor features with every spatio-temporal region in the scene to produce attention maps between the actor and the context. Girdhar et al. (2019), instead, proposed a Transformer-style architecture (Vaswani et al., 2017) to weight actors with features from the context around him. Finally, weakly-supervised approaches have also been proposed (Escorcia et al., 2020).

Graph-based representations. Graph-based representations have been used in action recognition (Brendel and Todorovic, 2011; Jain et al., 2016; Wang and Gupta, 2018; Zhang et al., 2019) to model spatio-temporal relationships, although the use of graph learning and graph convolutional neural networks (Defferrard et al., 2016; Kipf and Welling, 2017; Veličković et al., 2018) in video action detection is still under-investigated. Wang and Gupta (2018) proposed to model a video clip as a combination of the whole clip features and weighted proposal features, computed by a graph convolutional network based on similarities in the feature space and spatio-temporal distances between detections. Zhang et al. (2019), instead, defined the strength of a relation between two nodes in the graph as the inverse of the Euclidean distance between entities in the video.

3. Graph-based learning of spatio-temporal interactions

Given a video clip, the goal of our approach is to localize each actor and classify his actions. As actions performed in a clip depend on actor and object relationships through both space and time, we define a graph representation in which actor and object detections are treated as nodes, and edges hold relationships between them. Further, we link graphs from subsequent clips in time, to encode relations between clips belonging to the same longer video. We name our approach STAGE, as an acronym of *Spatio-Temporal Attention on Graph Entities*.

In this section, we first outline our graph-based representation for a single clip, describing the graph attention layer and the adjacency matrix we employ. In the remainder of the section, we will then extend this approach to handle a sequence of consecutive clips.

3.1. Graph-based clip representation

We propose a graph-based representation of each clip, where nodes consist of actor and object features predicted by pre-trained detectors, as shown in the left side of Fig. 2. Denoting the number of actors and objects belonging to clip t (*i.e.* centered in frame t) as A_t and O_t respectively, the total number of graph entities is $N_t = A_t + O_t$. Under this configuration, a clip can be represented as an $N_t \times d_f$ matrix, where d_f is the node feature size.

Since actors can have meaningful relations both between them and with objects in the scene, we employ a fully-connected graph representation, in which all nodes are connected to the others, as the input of our network. Following the assumption that the closer an entity is to another, the higher the probability that they affect each other, a link between two entities in the graph is made stronger if they are spatially close. The graph configuration is therefore given by a dense $N_t \times N_t$ adjacency matrix $\boldsymbol{A}$, in which $\boldsymbol{A}_{ij}$ is defined as the proximity between entities i and j, computed as follows:

$$\boldsymbol{A}_{ij} = e^{-\sqrt{(x_{ci}-x_{cj})^2+(y_{ci}-y_{cj})^2}}, \tag{1}$$

where x_{ci} and y_{ci} are the center coordinates of entity i. In the remainder of the paper, this single-clip adjacency matrix representation will be extended to a multi-clip adjacency matrix, allowing us to easily link graphs coming from subsequent clips of the same video.

3.2. Spatial-aware graph attention

Graph self-attention. Besides the introduced adjacency matrix, we adopt a graph attention module, inspired by Veličković et al. (2018). The input of the model consists of N_t node features which represent actors and objects, $\{\boldsymbol{f}_1, \boldsymbol{f}_2, \dots, \boldsymbol{f}_{N_t}\}$ with $\boldsymbol{f}_i \in \mathbb{R}^{d_f}$. First, the module applies a linear transformation to these features, in order to obtain a new representation of each entity $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_{N_t}\}$, $\boldsymbol{h}_i \in \mathbb{R}^{d_h}$. Then a *self-attention* operator $\mathcal{S}$ is applied to the nodes. In particular, the operator is defined as $\mathcal{S} : \mathbb{R}^{d_h} \times \mathbb{R}^{d_h} \to \mathbb{R}$, as follows:

$$\boldsymbol{E}_{ij} = \mathcal{S}(\boldsymbol{h}_i, \boldsymbol{h}_j) \tag{2}$$

with the scalar $\boldsymbol{E}_{ij}$ representing the *importance* of entity j with respect to entity i. Since we propose to represent a clip as a fully-connected graph, $\boldsymbol{E}_{ij}$ is computed for each pair of entities belonging to the same clip, avoiding the need for masking disconnected couples. Based on the original graph attention implementation (Veličković et al., 2018), $\mathcal{S}$ is implemented with a feedforward layer with $2 \times d_h$ parameters, followed by a LeakyReLU nonlinearity:

$$\boldsymbol{E}_{ij} = \text{LeakyReLU}(\text{FC}(\boldsymbol{h}_i \parallel \boldsymbol{h}_j)), \tag{3}$$

where $\parallel$ indicates concatenation on the channel axis and FC is a linear layer. The resulting matrix, $\boldsymbol{E}$, will be a squared matrix with the same shape as the adjacency matrix. Separating it into its components, it can be rewritten as:

$$\boldsymbol{E} = \begin{pmatrix} \boldsymbol{E}_{aa} & \boldsymbol{E}_{ao} \\ \boldsymbol{E}_{oa} & \boldsymbol{E}_{oo} \end{pmatrix} \tag{4}$$

where $\boldsymbol{E}_{aa}$ is the matrix of attentive weights between actors and actors, $\boldsymbol{E}_{ao}$ is the matrix of objects weights to actors, $\boldsymbol{E}_{oa}$ is the matrix of actors weights to objects and $\boldsymbol{E}_{oo}$ is the matrix of objects weights to objects.

Introducing spatial proximity. The proposed self-attention module, when applied to a clip graph, computes the mutual influence of two entities in feature space, *i.e.* the influence of an entity on another based on their features. However, it does not consider mutual distances between entities.

To introduce the prior given by the spatial proximity inside the clip, we condition the self-attention matrix $\boldsymbol{E}$ with the adjacency matrix $\boldsymbol{A}$, which contains the proximity between detections, by taking their Hadamard product, *i.e.*:

$$\boldsymbol{D} = \boldsymbol{A} \odot \boldsymbol{E}. \tag{5}$$

This operation allows us to strengthen the *importance* of the features of an entity with respect to its neighbors and to weaken relations between entities that lie spatially far from each other. A row-wise softmax normalization is then applied to obtain an importance distribution over entities:

$$\boldsymbol{W}_{ij} = \frac{\exp(\boldsymbol{D}_{ij})}{\sum_{k=1}^{N_t} \exp(\boldsymbol{D}_{ik})}. \tag{6}$$

The updated features computed by the module are a linear combination of the starting features $\{\boldsymbol{h}_1, \boldsymbol{h}_2, \dots, \boldsymbol{h}_{N_t}\}$ using $\{\boldsymbol{W}\}_{i,j}$ as coefficients. In particular, the self-attention module updates the initial features as follows:

$$\boldsymbol{h}'_i = \sigma\left(\sum_{j=1}^{N_t} \boldsymbol{W}_{ij} \boldsymbol{h}_j\right), \tag{7}$$

where σ is an ELU nonlinearity (Clevert et al., 2016).

3.3. Temporal graph attention

In this section, we extend the proposed attention-based approach to jointly encode a batch of consecutive clips. Since different clips can have a different number of actors and objects, we devise a single adjacency matrix with as many rows and columns as the total number of entities in all clips of the batch. Besides allowing us to manage clips with a variable number of entities, this solution is suitable to link more graphs together and avoids padding. When encoding a batch of consecutive clips, the size of the adjacency matrix will be $\sum_{t=1}^{b} N_t \times \sum_{t=1}^{b} N_t$, being b the size of the batch of clips.

An example is shown in Fig. 3, for a three clips setting and a temporal receptive field of three consecutive clips. Here, dark red elements contain the proximity between actors of the same clip, dark blue elements contain the proximity between objects of the same clip, and dark violet elements contain the proximity between actors and objects of the same clip. Entities belonging to subsequent clips can be linked by computing their boxes proximity (as in Eq. (1)), assuming that the temporal distance between clips is small enough to ensure the consistency of the scene. The light-colored elements of the adjacency matrix in Fig. 3 contain the proximity between actors (light red), objects (light blue), and actors/objects (light violet) belonging to two consecutive clips. The temporal receptive field of a single attentive layer can be potentially increased by adding the proximity of temporally distant entities in the adjacency matrix.

Self-attention over time. We extend the self-attentive operations to compute the *importance* of an entity with respect to all the other entities in the batch. In our implementation, the *self-attention* module computes attention weights for each pair of entity features, without any masking. For a three clips per batch setting, the complete attention weights matrix $\hat{\boldsymbol{E}}$ looks like the following:

$$\hat{\boldsymbol{E}} = \left(\begin{array}{ccc|ccc} \hat{\boldsymbol{E}}_{aa}^{t_0,t_0} & \hat{\boldsymbol{E}}_{aa}^{t_0,t_1} & \hat{\boldsymbol{E}}_{aa}^{t_0,t_2} & \hat{\boldsymbol{E}}_{ao}^{t_0,t_0} & \hat{\boldsymbol{E}}_{ao}^{t_0,t_1} & \hat{\boldsymbol{E}}_{ao}^{t_0,t_2} \\ \hat{\boldsymbol{E}}_{aa}^{t_1,t_0} & \hat{\boldsymbol{E}}_{aa}^{t_1,t_1} & \hat{\boldsymbol{E}}_{aa}^{t_1,t_2} & \hat{\boldsymbol{E}}_{ao}^{t_1,t_0} & \hat{\boldsymbol{E}}_{ao}^{t_1,t_1} & \hat{\boldsymbol{E}}_{ao}^{t_1,t_2} \\ \hat{\boldsymbol{E}}_{aa}^{t_2,t_0} & \hat{\boldsymbol{E}}_{aa}^{t_2,t_1} & \hat{\boldsymbol{E}}_{aa}^{t_2,t_2} & \hat{\boldsymbol{E}}_{ao}^{t_2,t_0} & \hat{\boldsymbol{E}}_{ao}^{t_2,t_1} & \hat{\boldsymbol{E}}_{ao}^{t_2,t_2} \\ \hline \hat{\boldsymbol{E}}_{oa}^{t_0,t_0} & \hat{\boldsymbol{E}}_{oa}^{t_0,t_1} & \hat{\boldsymbol{E}}_{oa}^{t_0,t_2} & \hat{\boldsymbol{E}}_{oo}^{t_0,t_0} & \hat{\boldsymbol{E}}_{oo}^{t_0,t_1} & \hat{\boldsymbol{E}}_{oo}^{t_0,t_2} \\ \hat{\boldsymbol{E}}_{oa}^{t_1,t_0} & \hat{\boldsymbol{E}}_{oa}^{t_1,t_1} & \hat{\boldsymbol{E}}_{oa}^{t_1,t_2} & \hat{\boldsymbol{E}}_{oo}^{t_1,t_0} & \hat{\boldsymbol{E}}_{oo}^{t_1,t_1} & \hat{\boldsymbol{E}}_{oo}^{t_1,t_2} \\ \hat{\boldsymbol{E}}_{oa}^{t_2,t_0} & \hat{\boldsymbol{E}}_{oa}^{t_2,t_1} & \hat{\boldsymbol{E}}_{oa}^{t_2,t_2} & \hat{\boldsymbol{E}}_{oo}^{t_2,t_0} & \hat{\boldsymbol{E}}_{oo}^{t_2,t_1} & \hat{\boldsymbol{E}}_{oo}^{t_2,t_2} \end{array}\right) \tag{8}$$

where $\hat{\boldsymbol{E}}_{aa}^{t,t'}$ is the weights matrix of actors belonging to clip t to actors belonging to clip t', $\hat{\boldsymbol{E}}_{ao}^{t,t'}$ is the weights matrix of objects belonging to clip t to actors belonging to clip t', and so on.

Given the new adjacency matrix, $\hat{\boldsymbol{A}}$, the Hadamard product:

$$\hat{\boldsymbol{D}} = \hat{\boldsymbol{A}} \odot \hat{\boldsymbol{E}} \tag{9}$$

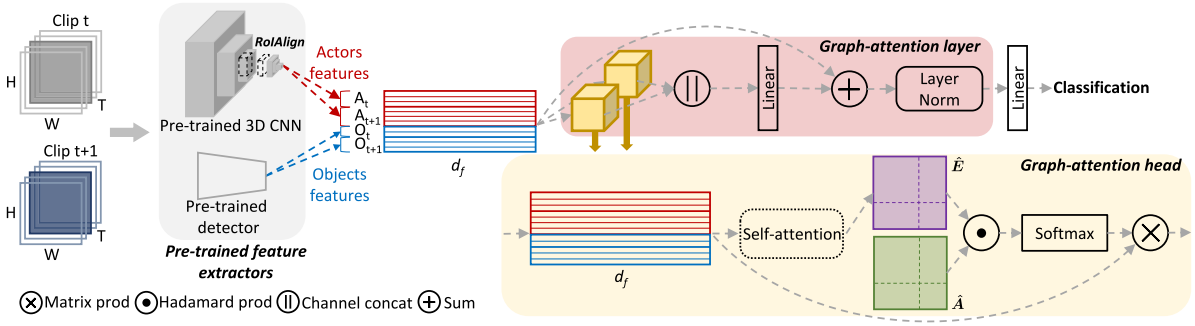

Fig. 2. Architecture of our high-level video understanding module. Given consecutive clips, the input of our model consists of actor and object features. It then applies a sequence of graph-attention layers (red-background box), each of them composed of a number of graph-attention heads (yellow-background box) applied in parallel. The figure depicts a single layer with two heads.

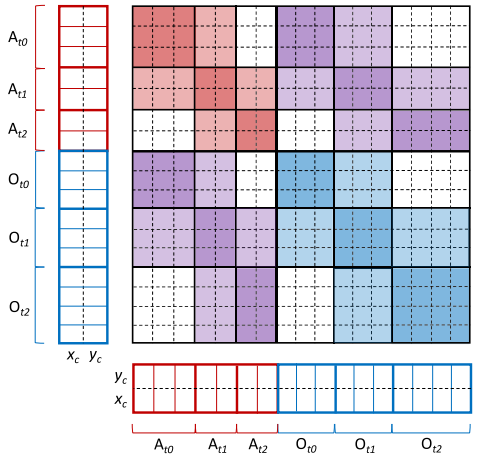

Fig. 3. Adjacency matrix in a three clips per batch configuration, containing the spatial proximity between entities belonging to the same clip (dark-colored sub-matrices) and to consecutive clips (light-colored sub-matrices). White elements are zeros. Bounding box centers are indicated as x_c and y_c.

is in charge of strengthening or weakening (based on the spatial distance) weights between entities belonging to the same timestamp or sufficiently close in time, and to zero weights between temporally distant entities. Finally, the linear combination of Eq. (7) replaces features of an entity with a weighted sum of features directly connected to it in the graph: these features come now from entities belonging to the same clip and to temporally close clips. As our approach works at the feature level, increasing the temporal receptive field of the module does not dramatically increase the demand of computational resources.

Multi-head multi-layer approach. A single graph-attention head (yellow-background box of Fig. 2) performs the aforementioned operations in our model. A graph-attention layer (red-background box of Fig. 2) concatenates the output of different heads and applies a linear layer, a residual connection, and a layer normalization (Ba et al., 2016). As it will be analyzed in Section 4.2, a grid search on the number of parallel heads and subsequent layers allows us to obtain the best performance.

It is worth noting that the number of graph attention layers affects the temporal receptive field of the overall sequence of layers. Considering a receptive field of three (corresponding to a graph where entities are directly connected only with other entities of the same clip and to entities from the previous and following clips), each layer after the first one increases the overall temporal receptive field by two. In a two layers setting, for instance, the second graph attention layer will compute the features of a clip as a weighted sum of its two neighbors, but features from those have already been affected by features of other clips in the first graph attention layer.

4. Experimental results

In this section, we introduce the experimental setting and report the implementation and training details. We then provide quantitative and qualitative evaluations, as well as computational analysis.

Datasets and metrics. We evaluate our model on versions 2.1 and 2.2 of the challenging AVA dataset (Gu et al., 2018), which contains annotations to localize people both in space and time and to predict their actions. It consists of 235 training and 64 validation videos, each 15 min long. The temporal granularity of annotations is 1 s, leading to 211k training and 57k validation clips centered in the annotated keyframes. Each actor is involved in one or more of 80 atomic action classes. One of the main challenges of AVA concerns its long-tail property: tens of thousands of samples are available for some classes while only a few dozen for others.

The performance of a model on AVA is measured by the keyframe-level mean Average Precision (mAP) with a 50% IoU threshold. Following the protocol suggested by the dataset authors and adopted in prior works, we train our architecture on all the 80 classes and evaluate its performance only on the 60 classes containing at least 25 validation examples.

We also report performances in terms of frame-mAP with the same 50% IoU threshold on two additional benchmarks, namely J-HMDB-21 (Jhuang et al., 2013) and UCF101-24 (Soomro et al., 2012). These two datasets are relatively smaller than AVA, and provide a single label per video. On average, they also have fewer interactions between entities.

People and object detectors. When experimenting on AVA, we use a Faster R-CNN (Ren et al., 2015) with a ResNeXt-101-FPN (He et al., 2016; Lin et al., 2017; Xie et al., 2017) as people detector, applied on keyframes. The detection network is pre-trained on COCO (Lin et al., 2014) and fine-tuned on AVA (Gu et al., 2018) people boxes. The detector is the same used by Feichtenhofer et al. (2019) and Wu et al. (2019), and reaches 93.9 AP@50 on the AVA validation set. Following previous works (Feichtenhofer et al., 2019; Gu et al., 2018), actor features are then obtained from a 3D CNN backbone (which is discussed in the next section) by replicating boxes in time to obtain a 3D RoI and applying RoIAlign (He et al., 2017). During training, we employ ground-truth people regions. Objects features, instead, are extracted from a Faster R-CNN detector pre-trained on Visual Genome (Krishna et al., 2017), and have a dimensionality of 2048.

Video Backbones. In all experiments, we employ a pre-trained actor backbone which is kept fixed during training. Freezing the backbone allows us to increase the batch size and to explore longer temporal relations between consecutive clips. The pre-trained backbones take raw clips as input and output features for each actor. When considering the AVA dataset, all video-level backbones are trained on the Kinetics dataset (Kay et al., 2017) and fine-tuned on the AVA dataset (Gu et al., 2018) before applying our module.

Table 1
Learnable blocks of STAGE, in a 4-heads/1-layer configuration, when using I3D features. GAL_i indicates the ith graph-attention-layer, which consists of 4 attention heads (each with 2 fully-connected layers), a linear layer, and a LayerNorm. N_t is the number of entities (actors and objects), A_t is the number of actors.

Stage	Module	Input size	Output size
Input		$N_t \times 1024$	
GAL_1	FC_{11}	$[N_t \times 1024] \times 4$	$[N_t \times 256] \times 4$
	FC_{12}	$[N_t \times N_t \times 512] \times 4$	$[N_t \times N_t] \times 4$
	FC_{13}	$N_t \times 1024$	$N_t \times 1024$
	LNorm	$N_t \times 1024$	$N_t \times 1024$
	FC_3	$A_t \times 1024$	$A_t \times classes$

On AVA, we consider three backbones for extracting actor features, namely I3D (Carreira and Zisserman, 2017), R101-I3D-NL (Wu et al., 2019) and SlowFast-NL (Feichtenhofer et al., 2019). The I3D (Carreira and Zisserman, 2017) backbone is pre-trained on ImageNet (Deng et al., 2009) before being "inflated", and then trained on Kinetics-400. RoIAlign is applied after the *Mixed_4f* layer and we fine-tune only the last layers (from the *Mixed_5a* layer to the final linear classifier) on AVA for 10 epochs. The R101-I3D-NL (Wu et al., 2019) and the SlowFast-NL (Feichtenhofer et al., 2019) backbones are pre-trained on Kinetics-400 or Kinetics-600 (R101-I3D-NL on ImageNet, too), and fine-tuned end-to-end on the AVA dataset.

For the I3D backbone, we use ground-truth boxes and predicted boxes with any score during features extraction, assigning labels of a ground-truth box to a predicted box if their IoU is 0.5 or more. We use predicted boxes with score at least 0.7 for the evaluation. Following the authors implementation (Feichtenhofer et al., 2019; Wu et al., 2019), for the R101-I3D-NL and SlowFast-NL backbones we use ground-truth boxes and predicted boxes with score at least 0.9 during features extraction, assigning labels of a ground-truth box to a predicted box if their IoU is 0.9 or more. We use predicted boxes with score at least 0.8 for the evaluation.

Features are always extracted from the last layer of the backbone before classification, after averaging in space and time dimensions: feature size, therefore, is 1024, 2048, and 2304 for I3D, R101-I3D-NL, and SlowFast-NL respectively. As additional features, we also add bounding boxes height, width and center coordinates to actors and objects, as we found it to be beneficial in preliminary experiments. A linear layer is employed to transform actor or object features to a common dimensionality d_f, making their concatenation feasible. In all experiments, when concatenating actor and object features we apply the linear layer to the feature vector with the largest dimensionality, leaving the other unchanged.

Implementation and training details. Each graph attention head consists of two fully-connected layers. The first one reduces the feature size depending on the number of heads used in that layer: with n_h heads, the output feature size is set to $\lfloor d_f / n_h \rfloor$. The second linear layer, instead, computes attention weights (Eq. (3)). The outputs of different attention heads are then concatenated, and a fully connected layer followed by a residual block and a layer normalization block is applied (Fig. 2). Each graph attention head is followed by a dropout with keep probability 0.5, and the alpha parameter of the LeakyReLU in Eq. (3) is set to 0.2. After a sequence of layers of the proposed module, one last linear layer is employed to compute per-class probabilities, and a sigmoid cross-entropy loss (for AVA) or a softmax cross-entropy loss (for the other benchmarks) is applied. In our experiments, we adopt a temporal receptive field of three, connecting entities of a clip with those belonging to the same, the previous and the following clip. Table 1 lists the learnable blocks of our architecture in a 4-heads/1-layer setting. It is worthwhile to mention that our graph-attention block is trained without any data augmentation, while end-to-end approaches typically require random flipping, scaling, and cropping.

Table 2
Comparison with previous approaches on AVA 2.1 validation set, in terms of mean Average Precision.

Model	Pretraining	mAP@50
AVA (Gu et al., 2018)	Kinetics-400	15.6
ACRN (Sun et al., 2018)	Kinetics-400	17.4
STEP (Yang et al., 2019)	Kinetics-400	18.6
Better baseline (Girdhar et al., 2018)	Kinetics-600	21.9
SMAD (Zhang et al., 2019)	Kinetics-400	22.2
RTPR (Li et al., 2018)	–	22.3
ACAM (Ulutan et al., 2020)	Kinetics-400	24.4
VATX (Girdhar et al., 2019)	Kinetics-400	24.9
SlowFast (Feichtenhofer et al., 2019)	Kinetics-400	26.3
LFB (R101-I3D-NL) (Wu et al., 2019)	Kinetics-400	26.8
I3D (Carreira and Zisserman, 2017)	Kinetics-400	19.7
STAGE (I3D)	Kinetics-400	**23.0**
R101-I3D-NL (Wu et al., 2019)	Kinetics-400	23.9
STAGE (R101-I3D-NL)	Kinetics-400	**26.3**
SlowFast-NL,8 × 8 (Feichtenhofer et al., 2019)	Kinetics-600	28.2
STAGE (SlowFast-NL,8 × 8)	Kinetics-600	**29.8**

Table 3
Comparison with previous approaches on AVA 2.2 validation and test sets, using different backbones. Numbers marked with "*" are obtained from models released in the official SlowFast (Feichtenhofer et al., 2019) repository. In (Feichtenhofer et al., 2019), the reported performances are 29.0 and 29.8 for SlowFast-NL,8 × 8 and SlowFast-NL,16 × 8 respectively.

Model	Pretraining	val mAP	test mAP
SlowFast-NL,8 × 8 (Feichtenhofer et al., 2019)*	Kinetics-600	29.1	–
STAGE (SlowFast-NL,8 × 8)	Kinetics-600	**30.0**	**29.6**
SlowFast-NL,16 × 8 (Feichtenhofer et al., 2019)*	Kinetics-600	29.4	–
STAGE (SlowFast-NL,16 × 8)	Kinetics-600	**30.3**	**29.9**
STAGE (SlowFast-NL,16 × 8,8 × 8)	Kinetics-600	**31.8**	**31.6**

During training, we use a batch size of 6. Adam optimizer (Kingma and Ba, 2015) is adopted in all our experiments, with a learning rate of 6.25×10^{-5} when using I3D features and 10^{-5} for R101-I3D-NL and SlowFast-NL features. The learning rate is decreased by a factor of 10 when the validation mAP does not increase for ten consecutive epochs. Early-stopping is also applied when the validation mAP does not increase for five consecutive epochs. All the experiments are performed on a single NVIDIA V100 GPU; on average, a single experiment takes less than a day to converge.

4.1. Main quantitative results

In the following experiments, we show that our proposed module can improve video action detection performance by a significant margin, reaching state-of-the-art results. If not specified otherwise, we employ a 2-layers 4-heads setting when using the I3D backbone and when testing on J-HMDB-21 and UCF101-24, and a 2-layers 2-heads setting when using the R101-I3D-NL and the SlowFast-NL backbones.

Results on AVA 2.1. Table 2 shows the mean Average Precision with 50% IoU threshold on AVA 2.1 for our method, considering the three backbones, and for a number of competitors. All the experiments refer to a single-crop validation accuracy (no multi-scale and horizontal flipping are adopted for testing) and for a single model (*i.e.*, without using ensemble methods).

When applying our approach, we observe a relative improvement of more than 16% on the I3D backbone (19.7 → 23.0), of about 10% for the R101-I3D-NL backbone (23.9 → 26.3) and of almost 6% for the SlowFast-NL backbone (28.2 → 29.8). Noticeably, the presence of non-local operations (Wang et al., 2018) in these last two backbones does not prevent our model from improving performance, underlying that these two techniques are complementary. Also, the results obtained using the I3D backbone are superior to many approaches that employ the same backbone and train end-to-end. The adoption of a long-term

Table 4
Mean Average Precision on different AVA 2.1 class groups (actions involving person–pose, person–person and person–object interactions), when using the I3D backbone.

Model	Pose	person–person	person–object
I3D (Carreira and Zisserman, 2017)	37.4	20.4	12.2
STAGE (I3D)	**40.4**	**23.5**	**15.7**
R101-I3D-NL (Wu et al., 2019)	41.4	26.5	15.5
STAGE (R101-I3D-NL)	**43.4**	**29.0**	**18.1**
SlowFast-NL (Feichtenhofer et al., 2019)	48.3	28.8	19.8
STAGE (SlowFast-NL)	**50.3**	**31.4**	**20.8**

Table 5
Experimental results on J-HMDB-21 and UCF101-24, using the detection backbones presented in Gkioxari and Malik (2015) and Saha et al. (2017). For both backbones, the three rows report the mAP presented in the original paper, the mAP we obtained training a linear classifier on top of pre-extracted features (marked by *) and the mAP of STAGE on the same features, respectively.

Model↓ Dataset→	J-HMDB-21	UCF101-24
Action tubes (Gkioxari and Malik, 2015)	27.0	–
Action tubes* (Gkioxari and Malik, 2015)	28.6	–
STAGE (Action tubes*)	**29.6**	–
AMTnet (Saha et al., 2017)	45.0	–
AMTnet* (Saha et al., 2017)	47.0	67.7
STAGE (AMTnet*)	**48.1**	**69.1**

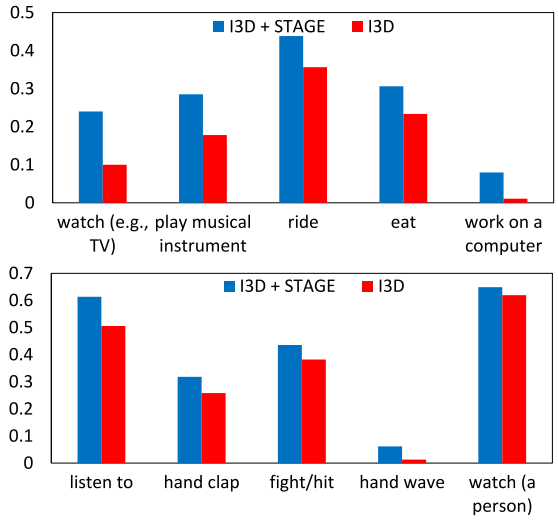

Fig. 4. Per-class Average Precision of an I3D backbone with and without our module. We report the five classes with the highest absolute gain among person–object interaction classes (top) and person–person interaction classes (bottom).

Table 6
Validation mAP obtained considering different combinations of graph-attention heads and layers.

Layers→	1			2			3		
Heads→	2	4	8	2	4	8	2	4	8
STAGE (I3D)	21.2	21.7	22.0	21.7	**23.0**	22.8	21.7	22.7	21.9
STAGE (R101-I3D-NL)	26.2	26.1	**26.3**	**26.3**	26.1	25.8	**26.3**	26.0	25.6
STAGE (SlowFast-NL)	29.7	29.2	29.6	**29.8**	29.3	29.6	29.6	29.3	29.5

feature bank in Wu et al. (2019) brings slightly better performance (26.8) compared to our solution (26.3) using the same R101-I3D-NL backbone. It is worth noting, although, that Wu et al. (2019) uses two instances of the backbone, one to compute long-term and another to compute short-term features, both fine-tuned end-to-end. Our model, instead, uses only one backbone instance, which is also kept fixed during training. Single-crop validation mAP obtained with the SlowFast-NL backbone (**29.8 mAP**) represents a new state of the art for the AVA v2.1 dataset.

Results on AVA 2.2. Table 3 reports the performance of our approach on the more recent AVA v2.2. Here, the test mAP has been computed using the official AVA evaluation server, after training on both train and val splits, following the common practice in literature. As it can be observed, adopting STAGE on top of SlowFast-NL,8x8 is better than doubling the number of input frames to the backbone (*i.e.*, using SlowFast-NL,16x8). This underlines that modeling high-level entities is at least as important as extracting better spatio-temporal features.

Finally, leveraging the fact that STAGE is backbone independent, we train it using both SlowFast-NL,8x8 and SlowFast-NL,16x8 features, which are averaged before being forwarded through STAGE. This model achieves single-crop **31.8 mAP**, a new state of the art for AVA v2.2.

Per-class analysis. In Table 4, we show the performances of our approach on different AVA class groups, *i.e.* actions involving person–pose (13 classes), person–person interactions (15 classes) and person–object interactions (32 classes). As it can be seen, our model shows a higher improvement for actions that involve an interaction between entities (which are also the majority in AVA). We also note that the recognition of pose classes (like *dance* or *martial art*) benefits from interactions and elements from the context. Finally, in Fig. 4, we show the five person–object interaction classes (top) and five person–person interaction classes (bottom) with the top AP gain, when considering the I3D backbone. Classes with the highest absolute gain are *watch (e.g., TV)* (+14.0 AP), *listen to (a person)* (+10.7 AP), *play musical instrument* (+10.7 AP), all involving interactions with other objects or actors.

Results on J-HMDB-21 and UCF101-24. We also experimented the capabilities of STAGE on two additional benchmarks, namely J-HMDB-21 (Jhuang et al., 2013) and UCF101-24 (Soomro et al., 2012), in comparison with the models presented in Gkioxari and Malik (2015) and in Saha et al. (2017). For fairness of evaluation, we adopt STAGE on top of the actor detection backbones presented in Gkioxari and Malik (2015) and in Saha et al. (2017). Hence, only actor boxes are considered, and no objects. J-HMDB-21 mAP is averaged over the three splits, while UCF101-24 mAP refers to the first split of the dataset, following the standard practice in literature. Results are reported in Table 5, where, for each dataset, we also report the frame-mAP obtained training a linear classifier on top of pre-extracted features (marked with *), and the frame-mAP obtained through STAGE applied on the same features. Only RGB features are considered, without exploiting optical flow. As it can be seen, in both settings applying STAGE leads to a performance improvement of around 1 mAP point.

4.2. Ablation study

To validate the importance of the design choices made in our graph-attention module, we run several ablation experiments. We explore different combinations of heads and layers, remove key components in the architecture, and modify the graph structure to use existing techniques in place of our choices. In all the following experiments we employ AVA v2.1.

Varying the number of heads and layers. Table 6 shows the effect of varying the number of graph-attention heads and layers when using STAGE with features from the three adopted backbones. As it can be noticed, stacking multiple layers together brings better performance: each layer after the first increases the temporal receptive field, generating indirect edges between temporally distant nodes. The best configuration is obtained when using a 4-heads/2-layers setting for I3D features and a 2-heads/2-layers setting for R101-I3D-NL and SlowFast-NL. These configurations are also used in the following ablation experiments.

Comparison with the STO baseline. In Table 7 we compare the STAGE module with the STO operator (Wu et al., 2019) applied on

Table 7
Comparison with the STO baseline.

Head↓ Backbone→	I3D	R101-I3D-NL	SlowFast-NL
1L STO	20.2	24.7	28.5
2L STO	20.4	25.0	28.7
STO (STAGE)	20.5	25.5	29.5
STAGE	**23.0**	**26.3**	**29.8**

Table 8
Validation mAP obtained removing some key components in our model or replacing them with other existing techniques.

Model↓ Backbone→	I3D	R101-I3D-NL	SlowFast-NL
Original Backbone	19.7	23.9	28.2
STAGE	**23.0**	**26.3**	**29.8**
STAGE w/o boxes proximity	21.5	25.8	29.6
STAGE w/o temporal connections	22.1	25.7	29.4
STAGE w/o actors–actors interactions	22.1	26.1	28.9
STAGE w/o object–object interactions	22.3	25.6	**29.8**
STAGE w/ Transformer attention	21.6	25.6	29.6
STAGE w/ nodes Euclidean distance	21.4	**26.3**	29.5

top of the considered backbones. The STO baseline consists of a short-term operator that updates actor features on the basis of other actors from the same clip, using one or more non-local blocks (Wang et al., 2018). STO lacks the graph structure, the object detections, and the temporal interactions, leading to worse performances compared to STAGE (Table 7 first two rows, corresponding to one-layer and two-layers STO, respectively). In the third row of Table 7, instead, we report the performance of the STAGE module when replacing Eq. (3) with the non-local-based attention, and removing temporal interactions. The original STAGE design demonstrates higher mAP in this case, too.

Removing key components. In Table 8 we report the validation mAP obtained when removing key components. Performances drop when removing the spatial prior between detections. Moreover, when the temporal links between consecutive clips are removed and only edges between nodes of the same clip are kept, we observe a reduction in mAP. To evaluate the design of the graph structure, we first remove actor–actor interactions to quantify the role of objects: in this setting, actor features are updated with a weighted sum of object features. As it can be seen, this leads to better performances compared to those of graph-free backbones. One can question if object–object interactions are useful: when removing them from the graph, performances drop for both I3D and R101-I3D-NL backbones. Our insight is that some recurring combinations of objects can be useful at prediction time: a closed door in clip t, for instance, related to the same open door in clip $t+1$ could help to recognize the *open* action.

Attention and adjacency alternatives. In the last two rows of Table 8 we show results obtained by replacing our attention mechanism and our adjacency matrix design with other proposals. We investigate the use of dot-product attention, by replacing the weights of Eq. (3) with weights computed through a Transformer-like self-attention (Vaswani et al., 2017), as follows:

$$\boldsymbol{E} = \frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\boldsymbol{V}, \tag{10}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ come from three linear projections of input features. In this setting, as it can be noticed, we again observe a significant drop in performance.

Taking inspiration from Zhang et al. (2019), we also replace the Euclidean distance between bounding box coordinates with the Euclidean distance between bounding box features in the adjacency matrix. We found this choice to lower the performance on both I3D and SlowFast-NL backbones. In this setting, Eq. (1) is replaced by:

$$\boldsymbol{A}_{ij} = \frac{1}{\|\boldsymbol{h}_i - \boldsymbol{h}_j\|_2}. \tag{11}$$

Table 9
Training times and computational requirements of STAGE and existing approaches involving an end-to-end finetuning of the 3D backbone. The LFB model uses two instances of the backbone, thus has twice the complexity of its base model.

Model	# GPUs	Clips/GPU	Epochs	Training time
ACRN (Sun et al., 2018)	11	1	63	–
Better baseline (Girdhar et al., 2018)	11	3	78	–
SMAD (Zhang et al., 2019)	8	2	11	–
VATX (Girdhar et al., 2019)	10	3	71	~7 days
SlowFast (Feichtenhofer et al., 2019)	128	–	68	–
LFB (Wu et al., 2019)	8 × 2	2	10 × 2	~2 × 2 days
STAGE	1	6	20	<1 day

Table 10
Computational complexity analysis for inference, considering 4 actors and 25 objects per clip.

Model	GFLOPs	Parameters
I3D (Carreira and Zisserman, 2017)	108	12M
+**STAGE**	+0.11	+6.4M
R101-I3D-NL (Wu et al., 2019)	359	54.3M
+**STAGE**	+0.24	+17M
SlowFast-NL,16 × 8 (Feichtenhofer et al., 2019)	234	59.9M
+**STAGE**	+0.26	+21.8M

It shall be noted that all the aforementioned ablations do not change the number of parameters in the model (except for the Transformer attention experiment, where each attention head uses three linear layers instead of two), thus confirming the effectiveness of our approach. We finally note that STAGE with SlowFast-NL,8 × 8 backbone reaches 36.5 validation mAP on AVA 2.1 when tested with ground-truth actor boxes, suggesting that a stronger person detector could significantly boost performances.

4.3. Computational analysis

Our module can reach state-of-the-art results without requiring end-to-end training of the backbone. This has an impact on the computational requirements of STAGE at training time, since the convolutional backbone incorporates most of the model complexity. Table 9 shows a comparison with different approaches employing end-to-end training in terms of training time and resource requirements. For each approach, we report the number of GPUs used during training, the batch size per GPU, the number of epochs, and the overall training time. The comparison is based on the implementation details reported in the original papers and refers to a training on the AVA (Gu et al., 2018) dataset. Our module requires a single GPU for training when pre-extracting backbone features, and less than a day to converge.

Finally, in Table 10, we report the additional FLOPs and trained parameters introduced by STAGE during inference. Please note that both the number of floating-point operations and the number of trained parameters depend on the dimensionality of the features produced by each backbone. As the amount of FLOPs also depends on the number of detections predicted on each clip, we consider a clip with a number of actor and object detections equal to the average number of actors and objects in all AVA training clips, *i.e.* 4 actors and 25 objects.

4.4. Qualitative analysis

We present some qualitative results obtained on clips of the AVA validation set in Fig. 5. Here, we only show the central keyframe of the clip; red and blue boxes represent predicted actors and objects respectively. For simplicity, we highlight only the actor involved in the action (despite other actors could be found in the scene), except for the *Kiss* class, where two actors perform the same action. Only predicted objects with score greater than 0.8 are shown, even if all detections are

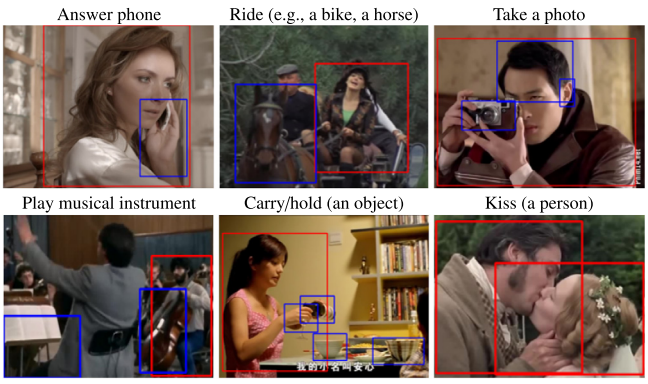

Fig. 5. Qualitative results showing the keyframes of the evaluated clips, with red boxes denoting actors performing actions, and blue boxes denoting objects.

Fig. 6. Sample failure cases. Actions are sometimes assigned to an actor very close to the one actually performing the action, and some object-interaction classes are wrongly assigned to people very close to the objects.

used during training. Fig. 6 shows sample failure cases. On average, we qualitatively observe that our spatio-temporal graph-based module is able to improve the recognition of human actions, especially for actions involving relationships with objects and other people.

5. Conclusion

In this work, we presented a novel graph-attention module that can be easily integrated into any video understanding backbone. The module computes updated actor features based on neighboring entities in both space and time. This is done considering detections from consecutive clips as a single learnable graph, where actors and objects are the nodes, while edges hold their relationships. The temporal distance between entities determines the presence of a direct edge between them, while the spatial distance and attention weights define its strength. Through extensive experiments on the Atomic Visual Actions (AVA) dataset and comparisons on J-HMDB-21 and UCF101-24, we showed that our module can bring state-of-the-art performances on a variety of backbones, thus highlighting the role of modeling high-level interactions in both space and time.

CRediT authorship contribution statement

Matteo Tomei: Conceptualization, Methodology, Software, Data curation, Writing - original draft. **Lorenzo Baraldi:** Conceptualization, Data curation, Writing - review & editing. **Simone Calderara:** Conceptualization, Methodology. **Simone Bronzin:** Funding acquisition, Project administration. **Rita Cucchiara:** Supervision, Writing - Review & Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially funded by Metaliquid Srl, Milan (Italy). We also acknowledge NVIDIA AI Technology Center for providing technical support and computational resources.

References

Ba, J.L., Kiros, J.R., Hinton, G.E., 2016. Layer normalization. arXiv preprint arXiv:1607.06450.

Bobick, A.F., Davis, J.W., 2001. The recognition of human movement using temporal templates. IEEE Trans. Pattern Anal. Mach. Intell. 23 (3).

Brendel, W., Todorovic, S., 2011. Learning spatiotemporal graphs of human activities. In: Proceedings of the International Conference on Computer Vision.

Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J., 2015. Activitynet: A large-scale video benchmark for human activity understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Carreira, J., Zisserman, A., 2017. Quo vadis, action recognition? a new model and the kinetics dataset In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Clevert, D.A., Unterthiner, T., Hochreiter, S., 2016. Fast and accurate deep network learning by exponential linear units (elus). In: Proceedings of the International Conference on Learning Representations.

Defferrard, M., Bresson, X., Vandergheynst, P., 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Adv. Neural Inf. Process. Syst..

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Escorcia, V., Dao, C.D., Jain, M., Ghanem, B., Snoek, C., 2020. Guess where? Actor-supervision for spatiotemporal action localization. Comput. Vis. Image Underst. 192.

Escorcia, V., Niebles, J., 2013. Spatio-temporal human-object interactions for action recognition in videos. In: Proceedings of the IEEE International Conference on Computer Vision Workshops.

Feichtenhofer, C., Fan, H., Malik, J., He, K., 2019. Slowfast networks for video recognition. In: Proceedings of the International Conference on Computer Vision.

Feichtenhofer, C., Pinz, A., Wildes, R., 2016a. Spatiotemporal residual networks for video action recognition. In: Advances in Neural Information Processing Systems.

Feichtenhofer, C., Pinz, A., Zisserman, A., 2016b. Convolutional two-stream network fusion for video action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Girdhar, R., Carreira, J., Doersch, C., Zisserman, A., 2018. A better baseline for AVA. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops.

Girdhar, R., Carreira, J., Doersch, C., Zisserman, A., 2019. Video action transformer network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Gkioxari, G., Malik, J., 2015. Finding action tubes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Gu, C., Sun, C., Ross, D.A., Vondrick, C., Pantofaru, C., Li, Y., Vijayanarasimhan, S., Toderici, G., Ricco, S., Sukthankar, R., et al., 2018. AVA: A video dataset of spatio-temporally localized atomic visual actions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Gupta, A., Davis, L.S., 2007. Objects in action: An approach for combining action understanding and object perception. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Gupta, A., Kembhavi, A., Davis, L.S., 2009. Observing human-object interactions: Using spatial and functional compatibility for recognition. IEEE Trans. Pattern Anal. Mach. Intell. 31 (10).

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn. In: Proceedings of the International Conference on Computer Vision.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Herath, S., Harandi, M., Porikli, F., 2017. Going deeper into action recognition: A survey. Image Vis. Comput. 60, 4–21.

Hou, R., Chen, C., Shah, M., 2017. Tube convolutional neural network (T-CNN) for action detection in videos. In: Proceedings of the International Conference on Computer Vision.

Hussein, N., Gavves, E., Smeulders, A.W., 2019. Timeception for complex action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Jain, A., Zamir, A.R., Savarese, S., Saxena, A., 2016. Structural-RNN: Deep learning on spatio-temporal graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Jhuang, H., Gall, J., Zuffi, S., Schmid, C., Black, M.J., 2013. Towards understanding action recognition. In: Proceedings of the International Conference on Computer Vision.

Kalogeiton, V., Weinzaepfel, P., Ferrari, V., Schmid, C., 2017. Action tubelet detector for spatio-temporal action localization. In: Proceedings of the International Conference on Computer Vision.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al., 2017. The kinetics human action video dataset. arXiv preprint arXiv:1705.06950.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations.

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: Proceedings of the International Conference on Learning Representations.

Kläser, A., Marszałek, M., Schmid, C., Zisserman, A., 2010. Human focused action localization in video. In: Proceedings of the European Conference on Computer Vision.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., et al., 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. Int. J. Comput. Vis. 123 (1), 32–73.

Laptev, I., 2005. On space-time interest points. Int. J. Comput. Vis. 64 (2–3), 107–123.

Li, D., Qiu, Z., Dai, Q., Yao, T., Mei, T., 2018. Recurrent tubelet proposal and recognition networks for action detection. In: Proceedings of the European Conference on Computer Vision.

Li, C., Zhong, Q., Xie, D., Pu, S., 2019. Collaborative spatiotemporal feature learning for video action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context. In: Proceedings of the European Conference on Computer Vision.

Prest, A., Ferrari, V., Schmid, C., 2012. Explicit modeling of human-object interactions in realistic videos. IEEE Trans. Pattern Anal. Mach. Intell. 35 (4).

Qiu, Z., Yao, T., Mei, T., 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In: Proceedings of the International Conference on Computer Vision.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems.

Saha, S., Singh, G., Cuzzolin, F., 2017. Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture. In: Proceedings of the International Conference on Computer Vision.

Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A., 2016. Hollywood in homes: Crowdsourcing data collection for activity understanding. In: Proceedings of the European Conference on Computer Vision.

Simonyan, K., Zisserman, A., 2014. Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems.

Soomro, K., Zamir, A.R., 2014. Action recognition in realistic sports videos. In: Computer Vision in Sports. Springer, pp. 181–208.

Soomro, K., Zamir, A.R., Shah, M., 2012. UCF101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402.

Sun, C., Shrivastava, A., Vondrick, C., Murphy, K., Sukthankar, R., Schmid, C., 2018. Actor-centric relation network. In: Proceedings of the European Conference on Computer Vision.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M., 2015. Learning spatiotemporal features with 3d convolutional networks. In: Proceedings of the International Conference on Computer Vision.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M., 2018. A closer look at spatiotemporal convolutions for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Ulutan, O., Rallapalli, S., Srivatsa, M., Torres, C., Manjunath, B., 2020. Actor conditioned attention maps for video action detection. In: The IEEE Winter Conference on Applications of Computer Vision, pp. 527–536.

Varol, G., Laptev, I., Schmid, C., 2017. Long-term temporal convolutions for action recognition. IEEE Trans. Pattern Anal. Mach. Intell. 40 (6), 1510–1517.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. In: Advances in Neural Information Processing Systems.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2018. Graph attention networks. In: Proceedings of the International Conference on Learning Representations.

Vezzani, R., Piccardi, M., Cucchiara, R., 2009. An efficient bayesian framework for on-line action recognition. In: 2009 16th IEEE International Conference on Image Processing. IEEE, pp. 3553–3556.

Wang, X., Girshick, R., Gupta, A., He, K., 2018. Non-local neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Wang, X., Gupta, A., 2018. Videos as space-time region graphs. In: Proceedings of the European Conference on Computer Vision.

Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L., 2016. Temporal segment networks: Towards good practices for deep action recognition. In: Proceedings of the European Conference on Computer Vision.

Wu, C.Y., Feichtenhofer, C., Fan, H., He, K., Krahenbuhl, P., Girshick, R., 2019. Long-term feature banks for detailed video understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Xie, S., Sun, C., Huang, J., Tu, Z., Murphy, K., 2018. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In: Proceedings of the European Conference on Computer Vision.

Xu, H., Das, A., Saenko, K., 2017. R-c3d: Region convolutional 3d network for temporal activity detection. In: Proceedings of the International Conference on Computer Vision.

Yang, X., Yang, X., Liu, M.Y., Xiao, F., Davis, L.S., Kautz, J., 2019. STEP: Spatio-temporal progressive learning for video action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Zhang, Y., Tokmakov, P., Hebert, M., Schmid, C., 2019. A structured model for action detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.