Universiteit Leiden

Master Computer Science

Learning Context for Weakly-Supervised Action Detection Using Graph Convolutional Networks

Name:	Michail Tsiaousis
Student ID:	s2082896
Date:	[07/07/2020]
Specialisation:	Data Science
1st supervisor:	Peter van der Putten (LIACS)
2nd supervisor:	Gertjan Burghouts (TNO)

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

This paper concerns the task of detecting and classifying human actions in video. The dominant paradigm for action detection is to learn spatio-temporal features using 2D or 3D Convolutional Networks. We argue that several actions are characterized by their context, such as relevant objects and actors present in the video. To this end, we introduce an architecture based on self-attention and Graph Convolutional Networks in order to model contextual cues, such as human-human and human-object interactions, so as to improve the classification of human actions in video for the task of action detection. We are interested in achieving this in a weakly-supervised setting i.e. using as less annotated data as possible in terms of action bounding box annotations. Furthermore, we investigate the ability of our model to explain its predictions by visualizing the learned contextual cues, encoded in the adjacency matrix, as an attention map. We show that our model is able to model relations between the actor and relevant context, even for actions unseen during training. Furthermore, we introduce an evaluation metric, based on recall, to quantitatively evaluate how well the attention highlights the important context. Our model relies on a 3D convolutional single (RGB) stream architecture, and does not require expensive optical flow computation. We evaluate our models on the challenging DALY dataset, which consists of human-object interaction actions and it is suitable for weakly-supervised action detection. Experimental results show that our contextualized approach outperforms a baseline action detection approach by more than 2 points in Video-mAP.

Acknowledgements

I would like to thank my university supervisor, Peter van der Putten, and TNO supervisors, Gertjan Burghouts and Fieke Hillerström, for their guidance and support during my thesis.

I would like to thank my beloved friends in Greece and Leiden. You have supported me during this journey more than you can imagine.

I would like to thank my family for their continuous, unconditional support.

1 Introduction

Human action recognition is an important part of video understanding, with potential applications in robotics, autonomous driving, surveillance, video retrieval and healthcare. Given a video, spatio-temporal action detection, also known as spatio-temporal action localization, aims to localize all human actions in space and time, and classify the actions being performed. With the success of Convolutional Neural Networks (CNNs) in various computer vision tasks [16, 19, 25, 27], the dominant paradigm is to extend CNN-based object detectors [15, 16, 28, 36] to learn appearance and motion representations in order to jointly localize and classify actions in video. Per-frame detections are linked throughout the video using tracking-by-detection [54] or tube-linking algorithms [5, 17, 39, 43] to form action tracks, or action tubes [17]: a sequence of bounding boxes connected in time that enclose the action.

In contrast to object detection, action detection requires learning of both appearance and motion features. This is achieved using two-stream 2D CNNs [42] operating on RGB and optical flow inputs, Long-Short Term Memory (LSTMs) networks [3, 21, 41] or 3D CNNs [6, 46, 57] and their two stream-variants, which perform convolutions in space and time. Spatio-temporal feature learning is essential to accurately localize and classify actions. Nevertheless, several actions share similar characteristics in terms of appearance and motion, which makes them difficult to differentiate. For example, consider the person performing the "Taking Photos" action in Figure 1. Learning only spatio-temporal features might be inadequate to differentiate this action from a similar one, such as "Phoning", since both share similar spatio-temporal characteristics (e.g similar posture, motion around the head). As humans, we make use of context to put actions and objects in perspective, which can be an important cue to improve action recognition. In Figure 1, context can refer to actor-object interactions; a person holding a camera is more likely to perform the action "Taking Photos" than "Phoning", and vice versa. Such interactions are typically encoded only implicitly by stacking several convolutional layers to capture a wider receptive field. In contrast, we are interested in directly modeling actor-actor and actor-object interactions.

In this paper, we introduce an approach to learn contextual cues, such as actor-actor and actor-object interactions, so as to aid action classification for the task of action detection. Our model, inspired by recent work on graph neural networks [26, 50], learns context by performing relational reasoning on a graph structure using Graph Convolutional Networks (GCN) [26]. The graph consists of context nodes and an actor node. The adjacency matrix, which encodes how relevant or important each context node is to the actor, is learned during training via gradient descent. Message passing in the graph in the form of weighted averaging of context node features adds context to the actor node. By adding relevant context to the actor, we expect to aid classification of the action being performed. A high-level overview is illustrated in Figure 1. Given a detected actor in a short video clip, we extract spatio-temporal features from the whole clip up to an intermediate convolutional layer. Every 1×1 spatio-temporal location of the output feature map corresponds to a context node in the graph. Features of the detected actor are converted to a fixed feature representation using RoI pooling [15] and spatio-temporal (3D) average pooling. Accordingly, the actor is represented by a single (gray) node in the graph. The adjacency matrix consists of relation values, indicating how relevant or important each context node feature is to the actor. Context is added to the actor via weighted averaging of context node features, weighted by their respective relation values.

We are interested in an approach to learn contextual cues using as less annotated data as possible. Action detection datasets such as J-HMDB [23] and AVA [18] lack object annotations, actor-actor and actor-object interaction labels. Therefore, recent works in action detection [13, 45, 47, 60] learn contextual cues in a weakly-supervised manner, requiring only actor bounding box annotations. However, they rely on full actor supervision during training, with annotations at every frame for J-HMDB, and at 1Hz for AVA which results in 1.5 million bounding boxes due to its immense size. Extensive video annotation is expensive, error-prone, and time consuming [30]. In this work, we are interested in learning context for the task of weakly-supervised action detection. Weakly-supervised action detection concerns the task of recognizing and detecting actions when a handful of annotated frames are

available throughout the video. Following the setting of sparse spatial supervision [55], we require up to five actor bounding box annotations throughout the action instance. Annotation takes place per actor, rather than per frame, that is, for multiple, concurrent actions, only one actor is annotated at a given frame. To this end, we use the Daily Action Localization in Youtube (DALY) [55] dataset annotated based on sparse spatial supervision. DALY consists of 10 action classes of human-object interactions (e.g. Drinking, Phoning, Brushing Teeth), which makes it a suitable testbed to model context for the task of weakly-supervised action detection.

In summary, we present an approach using Graph Convolutional Networks [26] to learn contextual cues, such as actor-actor and actor-object interactions, to aid action classification for the task of action detection. This is realized using weak supervision with respect to actor bounding box annotations and action labels. Our contributions are as follows:

1. We introduce an architecture employing Graph Convolutional Networks in order to model contextual cues so as to improve classification of human actions in videos.

2. Our model aids explainability by visualizing the graph's adjacency matrix in the form of attention maps, which highlight the learned context.

3. We achieve 1) and 2) when annotated data are sparse throughout the video, i.e. using up to 5 actor bounding box annotations per action instance.

4. We introduce an intuitive metric based on recall of retrieved objects in order to evaluate how well the model highlights the important context, such as objects relevant to the action.

5. We evaluate our model using the aforementioned metric on a zero-shot setting, namely, on actions and context unseen during training.

The paper is structured as follows. In Section 2, we present related work on action recognition, fully-supervised and weakly-supervised action detection, along with approaches on relational reasoning for action recognition. In Section 3, we present the baseline model and our approach on learning context by performing reasoning on a graph structure using Graph Convolutional Networks. Additionally, we discuss training of these models using sparse spatial supervision [55] and we provide implementation details. Section 4 presents the experimental setup, experiments and results. In Section 5, we perform a quantitative and qualitative analysis of our GCN model. In Section 6, we discuss similarities and differences of our method to related approaches, we reflect on the obtained results and we outline limitations of our approach and directions for future work. Finally, Section 7 concludes the paper.

2 Related Work

Our aim is to develop a model that reasons with respect to context so as to improve action classification for the task of action detection. Nevertheless, such a task usually requires a large amount of annotated data in the form of bounding boxes and class labels. In this paper, we are interested in modeling context when a handful of annotated frames are available throughout the video. In this section, we discuss related work on action recognition, fully-supervised and weakly-supervised action detection. Finally, we present related approaches on relational reasoning for action understanding.

2.1 Action Recognition

Action recognition aims to classify the action taking place in a video. Early approaches on action recognition employ two-stream architectures [42] operating at the frame level. The first stream receives RGB input to model appearance, and the second stream receives optical flow to model temporal information. Recent approaches operate on short video clips by directly performing spatio-temporal

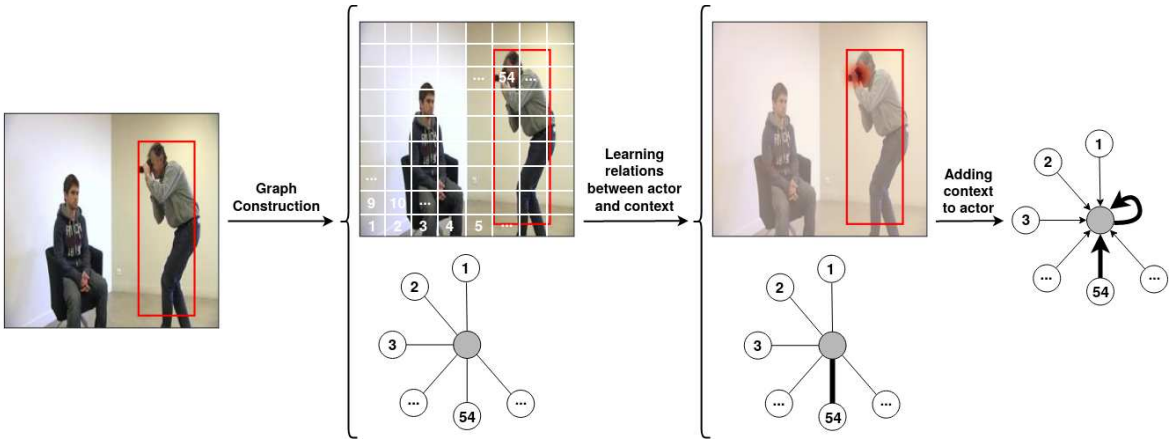

Figure 1: A high-level overview of our approach. Given a short input clip, features are extracted up to a convolutional layer. A graph is constructed on the resulting feature map with an actor (grey) node representing the detected actor and context nodes corresponding to every 1×1 spatio-temporal location. Our model performs relational reasoning on the graph to learn the relevant context for recognizing the action and accumulates contextual information to the actor node using graph convolutions.

(3D) convolutions [6, 46, 57]. The architecture is either one-stream (RGB), or two-stream (RGB and optical flow).

While action recognition considers the classification task, action detection carries out both classification and detection of the action(s) in a video. Next, we present related work on fully-supervised and weakly-supervised action detection.

2.2 Fully-Supervised Action Detection

Action detection is usually addressed in a fully-supervised setup. In this setting, an action detector is trained on action bounding box annotations and class labels, provided at every frame. Based on the two-stream architecture [42] for action recognition, early approaches [17, 35, 39, 43, 54, 59] rely on object detectors [15, 16, 28, 36] operating on RGB and optical flow inputs. Action classification and bounding box regression is performed at the frame level by fusing appearance and motion features generated by the the two streams. In [61], model complexity of two-stream networks is reduced by embedding RGB and optical flow representations into a single stream. Instead of operating at the frame level, Kalogeiton et al. [24] captures temporal context by stacking features extracted from a sequence of frames. Hou et al. [22] employ C3D [46] as a backbone network to model spatio-temporal features with 3D convolutions. To this end, they extend Region of Interest (RoI) pooling [15] and the Region Proposal Network (RPN) of Faster R-CNN [36] to operate on 3D volumes. Saha et al. [38] regresses and classifies micro-tubes from 3D region proposals spanning two successive frames. Their method is able to model temporal information without the use of optical flow. Finally, Gu et al. [18] extend Faster R-CNN [36] to operate on short video clips by incorporating the 3D CNN, I3D [6], as a backbone network for feature extraction.

2.3 Weakly-Supervised Action Detection

Deep learning classification and detection models require a large amount of annotated data. In fully-supervised action detection, annotation is in the form of action bounding boxes and corresponding class labels for every video frame. Clearly, annotation is expensive, time consuming and error prone [30].

Weakly-supervised action detection is used to train models when annotations are not provided at every frame. For action detection, weakly-annotated data might consist of a few annotated frames with bounding boxes and their class labels, class labels at the video level, or even binary class labels indicating the presence or absence of a class.

Sivan and Xiang [44] approach weakly-supervised action detection using Multiple Instance Learning (MIL). Their approach requires binary labels at the video level, indicating the presence of an action. In [12], action class labels at the video level are used as supervisory signal. The authors rely on a pre-trained detector and tracker to localize actors, and develop a novel actor-attention mechanism for action classification. Mettes et al. [30] propose action annotation using points, instead of boxes. A MIL algorithm is presented to select positive and negative proposals based on annotated points to train an SVM classifier. In a following work [29], the authors extend point-supervision to pseudo-annotations based on visual cues such as person detections and motion, among others. Their method assumes action labels at the video level. Chéron et al. [10] present a unified framework for action detection by incorporating varying levels of supervision in the form of labels at the video level, actions' temporal bounds, a few bounding boxes, etc.

Weinzaepfel et al. [55] introduce DALY and the setting of sparse spatial supervision, in which, up to five bounding boxes are available per action instance. Actors are detected using Faster R-CNN [36] and tracked throughout the video using a tracking-by-detection approach [54]. Tubes are classified using Fast R-CNN [15] and Improved Dense Trajectories (iDT) [51]. Chesneau et al. [11] produce full-body actor tubes inferred from detected body parts, even when the actor is occluded or part of the actor is not included in the frame.

2.4 Visual Relational Reasoning

Relational reasoning refers to reasoning with respect to relations between abstract entities, and performing inference for the task at hand based on these relations. There has been recent research on augmenting deep learning models with the ability to perform relational reasoning. In the following, we present approaches on relational reasoning for action understanding based on the relation network [40], attention mechanisms [9, 32, 34, 49] and Graph Convolutional Networks [26].

Santoro et al. [40] proposed the relation network which models relations between pairs of feature map pixels exhibiting state-of-the-art performance on visual question answering. This idea has been extended for action recognition to model object relations [4] and temporal dependencies [62] in successive video frames, and for action detection [45] so as to model relations between the actor and its surrounding scene, the latter representing the context.

The self-attention mechanism of the Transformer architecture [49] can also be seen as a form of relational reasoning, where an output is formed as a weighted sum of inputs, the weights being the attention or relation values. With regard to visual attention for action recognition, Girdhar et al. [14] present a novel attentional pooling layer, and non-local neural networks [52] compute the output of a feature map pixel as a weighted sum of all input pixels. For action detection, attention maps conditioned to each actor [47] provide contextual cues to improve action classification. Finally, Girdhar et al. [13] extend the Transformer architecture [49] for action detection.

When data are represented as graphs, relational reasoning with respect to a graph node corresponds to aggregating information from its local neighbors, with edge weights representing relation values. In this work, we apply Graph Convolutional Networks (GCN) [26], initially proposed for node classification on graphs. Recently, GCN have been used for visual relational reasoning for the tasks of action recognition [8, 53], group activity recognition [56], and action detection [60]. In these works, nodes represent detected actors and objects [53, 60], actors performing group activities [56], and convolutional features mapped to a lower dimensional space [8].

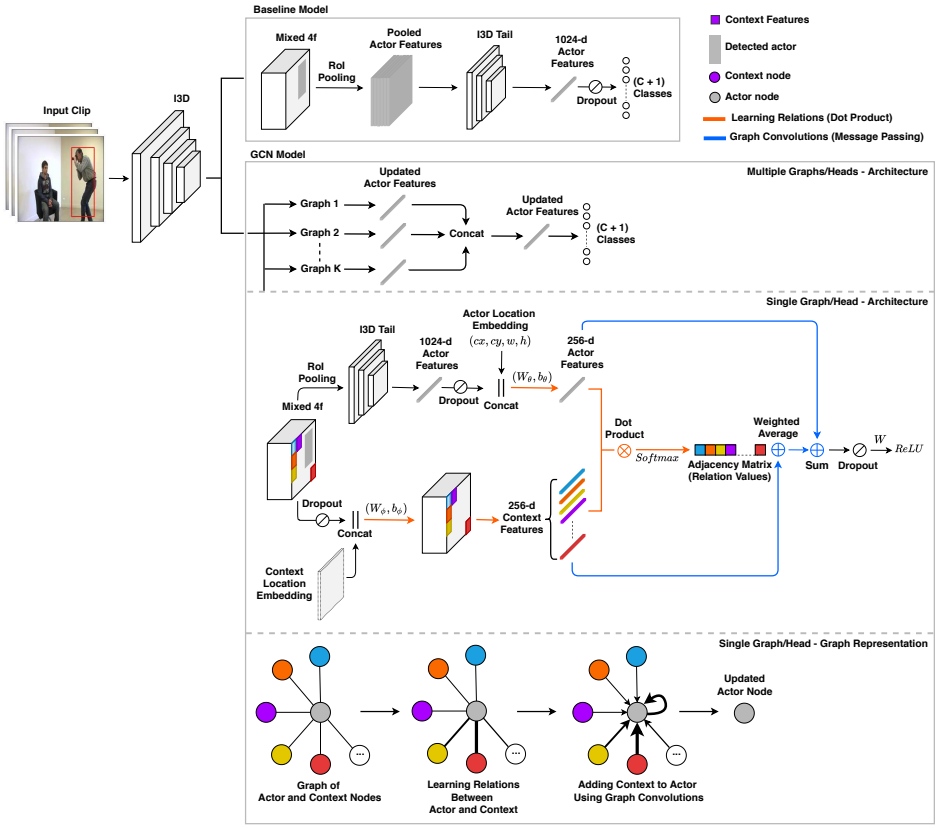

Figure 2: Method overview. The first branch depicts the baseline model. The second branch illustrates the proposed architecture based on Graph Convolutional Networks. The middle part of the GCN model shows the model architecture for a single graph, which is shown as a graph representation in the lower part. The top part of the GCN model illustrates the construction of multiple graphs in order to learn different types of context (Section 3.3.3).

3 Methods

We propose an approach to learn contextual cues, such as actor-actor and actor-object interactions, by performing relational reasoning between the actor and the context on a graph structure using Graph Convolutional Networks (GCN) [26]. These contextual cues are added to the actor feature representation so as to improve action classification, as we expect them to be discriminative for the action being performed. We compare the GCN model to a baseline model that uses no context, and classifies the action using the feature representation corresponding to the actor bounding box.

An overview of the baseline model and our proposed method is presented in Section 3.1, and they are described in detail in Section 3.2 and Section 3.3, respectively.

3.1 Method Overview

An overview of our baseline action detection model is shown in the first branch of Figure 2. The input is a short video clip with at least one actor performing an action. A 3D convolutional network extracts spatio-temporal features for the input clip, up to a convolutional layer. Actor features, i.e. features for each detected actor, are extracted by RoI pooling [15] on the resulting feature map, and they are passed though subsequent 3D convolutional layers and a spatio-temporal average pooling

layer. Finally, a linear layer outputs classification scores for C action classes and a background class.

We are interested in developing a model with the ability to reason with respect to contextual cues, such as interactions between different actors and objects. We expect such contextual cues to improve action classification, as they can be discriminative for the actions performed. The idea is to learn which parts of the feature map (context) are important to correctly recognize the action, and accumulate these contextual features to the actor features resulting in updated actor features.

To this end, we utilize GCN to model relations between the actor and the context. An overview of our method is illustrated in the second branch of Figure 2. The middle part of the second branch illustrates the model architecture for a single graph (see Section 3.3.3). The lower part provides a high level representation of the architecture using a graph structure. Similar to the baseline, a 3D convolutional network extracts features for the input clip. As the resulting feature map already encodes contextual information from the whole scene, we treat every 1×1 spatio-temporal location of the feature map as context. Actor features are extracted by RoI Pooling [15], and they are passed though subsequent 3D convolutional layers and a spatio-temporal average pooling layer. We construct a graph consisting of context nodes and an actor node, with connections drawn from every context node to the actor node. In order to accumulate context to the actor, we calculate a weighted average of the context features, weighted by their respective graph edge values. The original actor features are included in the graph by imposing an identity connection to the actor node. Following the self-attention mechanism [49, 50], the adjacency matrix, i.e. the graph edge values, is learned during training using a dot-product operation between context features and actor features. Accumulating context to the actor results in updated actor features, which are then used for classification of C action classes and a background class.

3.2 Baseline

Our baseline model consists of a 3D convolutional network augmented with a RoI pooling layer [15] to extract features for each actor for action classification. We use I3D [6] as a backbone network for feature extraction, which has exhibited impressive results on action recognition. The input to I3D is a sequence of frames of size $C \times T \times H \times W$, where C denotes the number of channels, and T is the number of input frames. H and W represent the height and width of the input sequence, respectively.

A sequence of T frames is passed through I3D which extracts features up to `Mixed_4f` layer. The size of the output feature map is of size $D' \times T' \times H' \times W'$, where D' denotes the number of feature channels, $T' = \frac{T}{8}$, $H' = \frac{H}{16}$, and $W' = \frac{W}{16}$. For each tube, a RoI pooling layer extracts a fixed feature representation of size $D' \times T' \times 7 \times 7$, encoding the feature representation of each actor. RoI pooling is performed independently on each of the T' temporal feature map dimensions. The actor features are then passed through I3D tail consisted of 3D convolutional layers `Mixed_5b` and `Mixed_5c`. Spatio-temporal (3D) average pooling is applied to the output, and the resulting representation is passed through a linear layer that outputs scores for C action classes and a background class.

3.3 Graph Convolutional Networks

In this section, we present our approach on learning contextual cues using GCN. In Section 3.3.1, we describe the graph construction, we define the nodes and edges of the graph, and present the mechanism of learning the adjacency matrix, i.e. the relations between the actor and the context. Section 3.3.2 presents the convolution operation on the graph structure, that is, information propagation on the graph and context accumulation to the actor. In Section 3.3.3, we extend our approach to multiple graphs in order to capture different types of context. In Section 3.3.4, we model spatial relations between the actor and the context by adding location information in the graph, a property which is lost when moving from regular convolutions to convolutions on a graph structure. In Section 3.4, we describe model training in a weakly-supervised setting using sparse spatial supervision [55]. Finally, Section 3.5 provides implementation details.

3.3.1 Graph Construction

Our graph consists of two types of nodes: context nodes and actor nodes. Context node features are obtained from the output feature map of `Mixed_4f` layer, and composes the set, $\mathbf{f}' = \{f'_1, f'_2, \ldots, f'_M\}$, where $f'_j \in \mathbb{R}^{D' \times 1 \times 1 \times 1}$, $M = T'H'W'$, and j indexes the $1 \times 1 \times 1$ spatio-temporal location of the feature map. Accordingly, every D'-dimensional feature of the output feature map acts as a context node in the graph. Actor nodes is the set, $\mathbf{a}' = \{a'_1, a'_2, \ldots, a'_N\}$, $a'_i \in \mathbb{R}^{D'' \times 1}$, where, N, is the number of detected actors in the input clip. The set of actor features, $\mathbf{a}'$, is obtained by RoI pooling [15] on the output of `Mixed_4f`, passing the pooled features through I3D tail, and then applying spatio-temporal (3D) average pooling.

Learning of relations between actor and context is shown in orange arrows in Figure 2. Relation or attention values are computed using a dot-product operation between each pair of actor features and context features. Prior to dot-product computation, a learnable linear transformation projects actor features and context features, respectively, into a lower dimensional space. During training, the model learns appropriate feature projections for the actor and the context, so that their dot-product outputs meaningful relations. Formally, $a_i = \mathbf{W}_\theta a'_i + \mathbf{b}_\theta$ and $f_j = \mathbf{W}_\phi f'_j + \mathbf{b}_\phi$, are transformations for the actor features and context features, respectively, with $\mathbf{W}_\theta \in \mathbb{R}^{D'' \times D}, \mathbf{W}_\phi \in \mathbb{R}^{D' \times D}$; $\mathbf{b}_\theta, \mathbf{b}_\phi \in \mathbb{R}^{D \times 1}$; $D < D', D''$. We denote the set of transformed context features by $\mathbf{f} = \{f_1, f_2, \ldots, f_M\}$, $f_j \in \mathbb{R}^{D \times 1 \times 1 \times 1}$ and transformed actor features by $\mathbf{a} = \{a_1, a_2, \ldots, a_N\}$, $a_i \in \mathbb{R}^{D \times 1}$. In matrix form, $\mathbf{F} \in \mathbb{R}^{M \times D}$ for context features, and $\mathbf{A} \in \mathbb{R}^{N \times D}$ for actor features.

The graph is represented by an adjacency matrix, $\mathbf{G} \in \mathbb{R}^{N \times M}$, where $g_{ij} \in \mathbf{G}$ denotes the relation or attention value, indicating the importance of context feature, f_j, to actor feature, a_i. Consequently, $\mathbf{G}$ is a directed graph connecting every context node to every actor node. The dot-product is given by

$$e_{ij} = a_i^T \cdot f_j \tag{1}$$

and relation values are obtained by applying a softmax normalization across context features

$$g_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})} \tag{2}$$

Softmax normalization stabilizes training by transforming the unnormalized relation values, $e_{ij}, i = 1, \ldots, N,\ j = 1, \ldots, M$, into a discrete probability distribution across context features, i.e. $0 \leq g_{ij} \leq 1,\ \sum_j g_{ij} = 1$.

3.3.2 Convolutions on Graphs

Having defined the graph structure, we perform reasoning on the graph using GCN. The output of the l^{th} layer can be expressed as

$$\mathbf{A}^{(l+1)} = \sigma\Big(\mathbf{Z}^{(l)}\mathbf{W}^{(l)}\Big) = \sigma\Big(\Big(\mathbf{G}^{(l)}\mathbf{F}^{(l)} + \mathbf{A}^{(l)}\Big)\mathbf{W}^{(l)}\Big) \tag{3}$$

where $\mathbf{Z}^{(l)} = \mathbf{G}^{(l)}\mathbf{F}^{(l)} + \mathbf{A}^{(l)}$, $\mathbf{Z}^{(l)} \in \mathbb{R}^{N \times D}$, $\mathbf{G}^{(l)} \in \mathbb{R}^{N \times M}$ is the adjacency matrix, $\mathbf{W}^{(l)} \in \mathbb{R}^{D \times D}$ is the weight matrix of a learnable linear transformation, and $\sigma(\cdot)$ is a non-linear activation function, such as ReLU. Message passing in the l^{th} layer is performed by $\mathbf{Z}^{(l)} = \mathbf{G}^{(l)}\mathbf{F}^{(l)} + \mathbf{A}^{(l)}$, and is shown in blue arrows in Figure 2. Namely, weighted average of $\mathbf{F}^{(l)}$ with the relation values, $\mathbf{G}^{(l)}$, produces weighted context features. Adding the actor features, $\mathbf{A}^{(l)}$, to the resulting representation imposes identity links for all actor nodes in the graph. $\mathbf{Z}^{(l)}$ is then passed through linear transformation, $\mathbf{W}^{(l)}$, and non-linear activation, $\sigma(\cdot)$, that outputs updated actor features, $\mathbf{A}^{(l+1)}$. The updated actor features can serve as an input to the next GCN layer, $l + 1$.

3.3.3 Building Multiple Graphs

A single graph may capture a single type of relation between the actor and the context. Building multiple graphs is equivalent to performing multi-head attention [49], which has shown to be effective in modeling different types of relations. To this end, we employ multi-head attention similar to [13, 49, 50, 53, 56] by constructing multiple graphs at a given layer and merging their outputs. This process is shown in the top part of the GCN model in Figure 2. Formally, we extend Equation 3 to operate on multiple graphs as follows

$$\mathbf{A}^{(l+1)} = \Big\Vert_{k=1}^{K} \sigma\Big(\mathbf{Z}^{(k,l)}\mathbf{W}^{(k,l)}\Big) = \Big\Vert_{k=1}^{K} \sigma\Big(\Big(\mathbf{G}^{(k,l)}\mathbf{F}^{(k,l)} + \mathbf{A}^{(k,l)}\Big)\mathbf{W}^{(k,l)}\Big) \tag{4}$$

where K is the number of graphs in the l^{th} layer, and $\|$ denotes concatenation. Weight matrices and bias terms for dot-product computation, $\mathbf{W}_\theta^{(k,l)}, \mathbf{b}_\theta^{(k,l)}, \mathbf{W}_\phi^{(k,l)}, \mathbf{b}_\phi^{(k,l)}$, and layer weight matrices, $\mathbf{W}^{(k,l)}$, are independent across multiple graphs, i.e. their parameters are not shared.

3.3.4 Location Embedding

Location information, such as the position of an actor with respect to other actors and objects, is important for modeling contextual cues. As an example, the "Drinking" action consists of object manipulation around the head, and actors or objects located far away from the actor of interest provide, possibly, less important context.

Location information encoded indirectly by regular convolutions is lost when represented as a graph structure. Note that the colored context features in Figure 2, i.e. the context nodes of the graph, do not posses any specific order or position to associate them with their spatial location in the output feature map.

We incorporate location information to both context features and actor features. For context features, we concatenate coordinates (x, y) along the channel dimension of the feature map, prior to applying $\mathbf{W}_\phi$. Coordinates indicate the spatial location of each 1×1 cell on the feature map, and they are normalized between $[-1, 1]$. For actor features, we concatenate vector (cx, cy, w, h) before applying $\mathbf{W}_\theta$, corresponding to the average center, width and height of the actor tube across the input clip. Actor coordinates are normalized between $[-1, 1]$. We expect location information to aid transformations $\mathbf{W}_\theta$ and $\mathbf{W}_\phi$ in learning better feature representations, so that, in turn, the dot-product of the resulting vectors produces more meaningful relations between the actor and the context.

3.4 Sparse Spatial Supervision

We are interested in learning context for action detection using only a handful of annotated frames per action instance. For an annotated frame, annotation is in the form of an action bounding box and corresponding class label. Additionally, annotation takes place per actor, rather than per frame, that is, for multiple, concurrent actions, only one actor is annotated at a given frame. To this end, we train our models using sparse spatial supervision as introduced in [55].

Tubes are labeled using sparse spatial supervision as follows. Actors are detected using an external detector, and then tracked throughout the video using a tracking-by-detection approach. Actor tubes are labeled based on spatio-temporal Intersection over Union (IoU) with sparse annotations, i.e. ground truth tubes comprised of up to 5 bounding boxes. In detail, we label each tube by computing the spatio-temporal IoU between the predicted tube and the ground truth tube in the action's temporal interval. The spatio-temporal IoU is defined as the average spatial IoU between the annotated boxes and the corresponding detections in the tube [55]. Tubes with spatio-temporal IoU greater than 0.5 are assigned to the action class of the ground truth tube with the highest IoU. Tubes are labeled as background when the spatio-temporal IoU with all ground truth tubes is less than 0.5

After extracting features for the input clip using I3D, the boxes of each tube corresponding to the input clip are appropriately scaled and mapped to the feature map produced by `Mixed_4f` layer. Boxes are mapped with a temporal stride of four frames, resulting in T' boxes for each tube, which is equal to the temporal dimension of the output feature map of `Mixed_4f` layer. Actor features are extracted by applying RoI Pooling independently on each temporal dimension, followed by I3D tail and spatio-temporal average pooling.

3.5 Implementation Details

In this section, we provide implementation details with respect to model architecture, weight initialization, training and inference procedure. We implement our models in PyTorch [33].

3.5.1 Architecture

I3D is pre-trained on ImageNet [37] and then on Kinetics [6] action recognition dataset. Input is a clip of 32 (T) RGB frames with spatial resolution of 224×224 ($H \times W$). The output of `Mixed_4f` layer is $832 \times 8 \times 14 \times 14$ ($D' \times T' \times H' \times W'$). Accordingly, every context feature is of size $832 \times 1 \times 1 \times 1$. The actor feature obtained by RoI pooling has dimensions $832 \times 8 \times 7 \times 7$. Passing them through I3D tail results in dimensions $1024 \times 8 \times 7 \times 7$ ($D'' \times T' \times H' \times W'$) , and spatio-temporal average pooling reduces the size to $1024 \times 1 \times 1 \times 1$. Transformations, $\mathbf{W}_\theta$, $\mathbf{W}$, are implemented as fully connected layers, and, $\mathbf{W}_\phi$, as a 3D convolutional layer with kernel size $1 \times 1 \times 1$. The output dimension is set to $D = 256$. We apply 3-dimensional dropout to context features before applying $\mathbf{W}_\phi$. Additionally, 1-dimensional dropout is applied to actor features before applying the actor transformation of the first GCN layer, $\mathbf{W}_\theta^{(k,1)}, k = 1, 2, \ldots, K$, and before $\mathbf{W}$ in all GCN layers. Finally, dropout is applied prior to the final linear layer in both GCN and baseline model. Dropout probability is 0.5 in all cases.

3.5.2 Weight Initialization

All fully connected layers are initialized using a Normal distribution with standard deviation, $std = \frac{gain}{\sqrt{fan\_in}}$, according to [20], where fan_in refers to the layer's input dimension. We set the gain parameter to 1 for $\mathbf{W}_\theta$ and to $\sqrt{2}$ for the rest of the fully connected layers. Biases are initialized to zero.

3D convolutional layer, $\mathbf{W}_\phi$, is initialized using a Uniform distribution according to [20] in the range $(-b, b)$, where $b = gain \cdot \sqrt{\frac{3}{fan\_in}}$, with $gain = \frac{1}{\sqrt{3}}$ and $fan\_in = D'$. Biases of $\mathbf{W}_\phi$ are initialized to zero. We found useful to slightly reduce the range for the initialization of $\mathbf{W}_\phi^{(k,1)}, k = 1, 2, \ldots, K$, i.e. context transformation in the first GCN layer. In practice, we use $(-b + 0.01, b - 0.01)$ in the first GCN layer and $(-b, b)$ in subsequent layers. We motivate this choice by the fact that the actor transformation in the first layer, $\mathbf{W}_\theta^{(k,1)}$, is initialized with standard deviation, $\frac{1}{\sqrt{D''}}$, which is smaller than the standard deviation of the same transformation in the second layer, $\frac{1}{\sqrt{K \cdot D}}$. The inequality holds for $D'' = 1024, K \leq 3$ and $D = 256$. Since the dot-product of the output of the context transformation and actor transformation produces the adjacency matrix (after a softmax normalization), we argue that the initialization of the context transformation may affect the weights of the actor transformation during training, and vice versa.

3.5.3 Detector

As described in Section 3.4, we rely on an external object detector to detect and track each actor throughout the video clip. In this paper, we use tubes provided by [55]. A Faster R-CNN [36] pre-trained on MPII Human Pose dataset [1] detects all actors in each frame, and detections are tracked throughout the video using a tracking-by-detection approach. Readers are referred to [54, 55] for a description of the tracking-by-detection approach.

3.5.4 Training and Inference

Models are optimized using SGD and cosine learning rate annealing schedule. Cosine annealing reduces the learning rate from a maximum value, max_lr, to a minimum value, min_lr, for $total\_epochs$ number of epochs, following a cosine curve. For the baseline model, we use $max\_lr = 2.5 \cdot 10^{-4}$ and $min\_lr = 0$, for a total of 150 epochs. For GCN, we use $max\_lr = 4.7 \cdot 10^{-5}$ and $min\_lr = 0$, for 450 epochs.

For GCN, we have also experimented with linear learning rate warm-up, in which, the learning rate increases linearly from an initial value, $init\_lr$, to max $\_lr$ for $warmup\_epochs$ number of epochs, and then follows a cosine curve. The intuition of using warm-up is that during the early stages of training, a small learning rate might prove helpful in mitigating noisy changes in the adjacency matrix, resulting from the random initialization of layers $\mathbf{W}_\theta$ and $\mathbf{W}_\phi$. Such noisy changes might lead the model to converge to a bad local minimum in later stages of training. When warm-up is used, we set $init\_lr = \frac{max\_lr}{10}$ and $warmup\_epochs = 3$ (1461 iterations).

We train our models by randomly sampling a 32-frame clip from each video in the training set. Tubes of each clip are scored using the softmax scores produced by the model. Models are trained using a batch size of 3 clips. Training time is approximately one day for GCN and less than half a day for the baseline on a GTX 1080 Ti GPU.

During inference, we sample 10 32-frame clips from each video, and tubes are scored by averaging the softmax scores across the clips. The same clips are sampled in order to facilitate fair comparison between different models. Finally, we apply per-class non-maximum-suppression (NMS) on the scored tubes of each action instance. NMS discards a tube if the latter has an IoU greater than a threshold with a higher scoring tube of the same class. NMS threshold is set to 0.2.

4 Experiments

In this section, we conduct experiments in order to evaluate the baseline and GCN model. In Section 4.1, we present the DALY dataset and the evaluation metric used throughout the experiments. Section 4.2 presents results for the baseline model, which is trained five times so as to examine model stability in terms of variance across different repetitions. In Section 4.3, we evaluate the effect of using more than one GCN layer and building multiple graphs per layer. Additionally, we consider two functions to merge the output of multiple graphs. The addition of linear learning rate warm-up is studied in Section 4.4. In Section 4.5 and Section 4.6, we evaluate the effect of removing the location embedding and I3D tail, respectively. The impact of weight initialization is studied in Section 4.7, and in Section 4.8 we compare our models with the state-of-the-art on DALY. Finally, in Section 4.9, we report results when minimum spatial supervision, i.e. one actor bounding box annotation per action instance, is used during training and inference.

We report results for Section 4.8 and Section 4.9 on the test set of DALY. Results for the rest of the experiments are reported on the validation set.

4.1 Dataset and Evaluation Metric

We develop and evaluate our models on the Daily Action Localization in Youtube (DALY) [55] dataset. It consists of 510 videos of 10 human actions, such as "Drinking", "Phoning", or "Brushing Teeth". It is suitable for temporal action localization, and therefore, out of 3.3 million frames (31 hours), approximately 700.000 of them contain an action. In this paper, we do not perform temporal localization, and we assume that the temporal boundaries of each action instance within a video are known. An action instance may contain more than one person performing an action. Each of the 10 actions classes contains an interaction between a person and an object that define the action taking place. For example, it is common for the actor to use a cup or a glass during the "Drinking" action. Hence, DALY is a suitable dataset for contextual modeling with respect to actor-object and actor-actor interactions. There are 31 training videos and 20 test videos per class. In order to fine-tune our models, we hold out a subset of the training set as a validation set, consisted of 10 videos from each class.

Model	Val. mAP
Baseline	46.8 (± 0.928) (47.79)

Table 1: Mean mAP, standard deviation and maximium mAP on the validation set across five repetitions of the baseline model.

We evaluate our models based on mean Average Precision (mAP) at the video level at IoU threshold, δ (Video-mAP@δ) [17]. Video-mAP is calculated as the mean of Video-AP. Video-AP measures, for each class, the area under the precision-recall curve of the tube detections. A detection is a true positive if the spatio-temporal IoU between the detected tube and ground truth tube is above threshold δ and the action is correctly classified. We report results for IoU threshold, $\delta = 0.5$.

4.2 Baseline

We train the baseline model five times in order to examine the variance in performance across different repetitions. We report in the following order, mean mAP, standard deviation and maximum mAP. Results are shown in Table 1. We obtain a mean mAP of 46.8 with a standard deviation of 0.928 and a maximum mAP of 47.79.

4.3 Number of Layers and Graphs

Our GCN model can employ multiple GCN layers by providing the output of one layer as input to the next layer. By increasing the number of GCN layers, we expect to encode contextual information on a higher level. Additionally, in order to capture different types of context, we perform multi-head attention by building multiple graphs in each layer (Section 3.3.3).

We experiment with up to two layers, and up to three graphs per layer. Additionally, we compare concatenation and summation as merging functions in order to combine the output of multiple graphs in the last GCN layer. Irrespective of the choice of merging function, for a two-layer GCN model, the first layer always uses concatenation. We provide the number of model parameters for every configuration (choice of merging function and number of layers and graphs). I3D parameters are not included, as they remain constant across different configurations.

Results are shown in Table 2. We observe an increase in mAP by moving from one graph to two graphs per layer, for both merging functions. For concatenation and two GCN layers, mAP increases even for three graphs per layer. Regarding the number of layers, it is interesting that mAP increases for a 2-layer GCN model with concatenation as merging function, but not with summation. Finally, concatenation outperforms summation in nearly all configurations.

For the rest of the experiments, we choose a 2-layer, 2-graph GCN model with concatenation as merging function. Although the best performance in terms of mAP is achieved using a 2-layer, 3-graph GCN model, we argue that two layers and two graphs per layer provide a good balance between performance and number of model parameters.

4.4 Learning Rate Warm-Up

In the following, we include linear learning rate warm-up during training in order to examine its effect on model performance. We perform five repetitions with warm-up, and without warm-up, of a 2-layer, 2-graph GCN model with concatenation as merging function. We report in the following order, mean mAP, standard deviation and maximum mAP.

Results are shown in Table 3. We obtain similar performance with respect to mean mAP, while the standard deviation is approximately two times larger when warm-up is added. On the other hand, warm-up results in greater maximum mAP (53.14). We conducted two two-sided significance tests, t-test and Mann-Whitney-Wilcoxon test, to compare the mAP means of the two models on

# Layers	# Graphs	Merging Function	# Parameters	Val. mAP
1	1	-	543K	49.39
1	2	Sum	1.084M	51.39
		Concat	1.086M	51.3
1	3	Sum	1.624M	50.7
		Concat	1.630M	50.98
2	1	-	887K	47.28
2	2	Sum	1.903M	50.1
		Concat	1.906M	51.82
2	3	Sum	3.050M	49.98
		Concat	3.055M	52.09

Table 2: Validation mAP with respect to different number of layers, number of graphs per layer and merging functions for combining the output of multiple graphs. The number of model parameters are provided for every configuration.

a significance level of 0.05. The tests did not result in significant evidence for a difference in mAP means, with p-values 0.887 and 0.841, respectively. Due to the large standard deviation in mAP, we opt for a model without warm-up for the rest of the experiments.

4.5 Location Embedding

We evaluate model's performance when the location embedding is not included in the architecture. To this end, we remove the coordinates for both context features and actor features. Accordingly, the model has no information of the actor's location relatively to other actors and objects. Relations between the actor and the context are calculated based solely on visual features.

Table 3 presents the results. By removing the location embedding, performance decreases by approximately 1.1 point in terms of mAP. This result indicates that encoding spatial information is beneficial for learning meaningful relations between the actor and the context.

4.6 I3D Tail

In this section, we examine the effect of I3D tail, i.e. 3D convolutional blocks `Mixed_5b` and `Mixed_5c`, on model performance. Accordingly, we remove the I3D tail from model's architecture, and actor features are extracted by applying RoI Pooling and spatio-temporal average pooling on the output of `Mixed_4f` layer. Note that the output of `Mixed_4f` is still used to encode the context features.

Results in Table 3 highlight the importance of using the I3D tail to encode the actor features. Removing the I3D tail results in a decrease of more than four points in mAP (47.42). Interestingly, this result is close to the one achieved by the baseline model (Table 1).

4.7 Weight Initialization

We experiment with different weight initializations for transformation, $\mathbf{W}_\phi$, the 3D convolutional layer that produces the transformed context features prior to dot-product computation. We use normal and uniform initialization with gain equal to 1 and $\frac{1}{\sqrt{3}}$. Additionally, we use the adjustment of reducing the range of the uniform distribution for the first GCN layer, as described in Section 3.5.2.

Results are presented in Table 4. For normal initialization, the choice of gain results in almost two points difference in mAP. It is interesting that, for uniform initialization, the choice of gain by itself

Warm-up	Loc. emb.	I3D tail	Val. mAP
	✓	✓	51.22 (± 0.55) (51.82)
✓	✓	✓	51.31 (± 1.05) (53.14)
		✓	50.7
	✓		47.42

Table 3: Validation mAP with respect to different model configurations including learning rate warm-up, location embedding and I3D tail.

Weight Init. ($\mathbf{W}_\phi$)	Gain	Val. mAP
Normal	1	51.52
	$\frac{1}{\sqrt{3}}$	49.78
Uniform	1	50.19
	$\frac{1}{\sqrt{3}}$	50.28
Uniform w/ adjustment	1	49.83
	$\frac{1}{\sqrt{3}}$	51.82

Table 4: Validation mAP for different initializations and gains of the context transformation, $\mathbf{W}_\phi$. "Adjustment" refers to the adjusted initialization described in Section 3.5.2.

does not affect performance. Nevertheless, the combination of $gain = \frac{1}{\sqrt{3}}$ with the proposed adjustment is comparable with normal initialization ($gain = 1$) and outperforms all other initializations.

We conclude that, firstly, weight initialization affects the final performance of the GCN model. Secondly, weight initialization might be an important architecture choice when learning contextual cues for action detection. Note that weight initialization has not been explored in related work [13, 45, 47, 60].

4.8 Comparison with the State of the Art

We compare the GCN model with the baseline model and the state-of-the-art on DALY [11, 55]. For GCN and baseline model, we report results on the validation set and test set across five repetitions. Finally, we report mAP results on the test set for [11, 55].

It is worth noting that our method is not directly comparable to [11, 55] for two reasons. Firstly, our models are trained using fewer videos, since we hold out a part of the training set as a validation set for fine-tuning. Secondly, although we use tubes provided by [55], these tubes are used in [11, 55] to label the region proposals produced by the detector (Section 3.5.3), as positive or negative. A Fast R-CNN [15] is then trained to classify these proposals. In contrast to [11, 55], we do not have access to the region proposals, so we classify directly the tubes produced by the detector. Consequently, we train our models using far less number of detections, since the RPN of Faster R-CNN [36] produces a large number of proposals which are then filtered by classifying them to one of the action classes or background, and by performing per-class NMS.

Results are shown in Table 5. We observe a large difference in validation and test mAP, which is apparent in both the baseline and GCN model. Using a different validation split resulted in a similar gap, therefore, we believe that the large difference is due to validation set size, which comprises 20% of the dataset, while the training set and test set size is 40%, respectively.

The GCN model outperforms the baseline model in terms of mAP in both validation set and test set. In the validation set, we obtain an absolute increase of 4.42 points in mean mAP and 4.03 points in maximum mAP, a relative increase of 9.44% and 8.43%, respectively. In the test set, the

Model	Architecture	Input	Val. mAP	Test mAP
* Weinzaepfel et al. [55]	Fast R-CNN (VGG-16)	RGB, OF	-	61.12
* Chesneau et al. [11]	Fast R-CNN (VGG-16)	RGB, OF	-	63.51
Baseline (Ours)	I3D	RGB	46.8 (± 0.928) (47.79)	59.58 (± 0.22) (59.79)
GCN (Ours)	I3D	RGB	51.22 (± 0.55) (51.82)	61.82 (± 0.51) (62.73)

Table 5: Comparison of the GCN model with the baseline model and with the state-of-the-art on the validation set and test set of DALY. Additionally, we report model architecture and input modality (OF stands for Optical Flow). We mark approaches that are not directly comparable to the GCN model with an asterisk (*); refer to Section 4.8 for details.

.

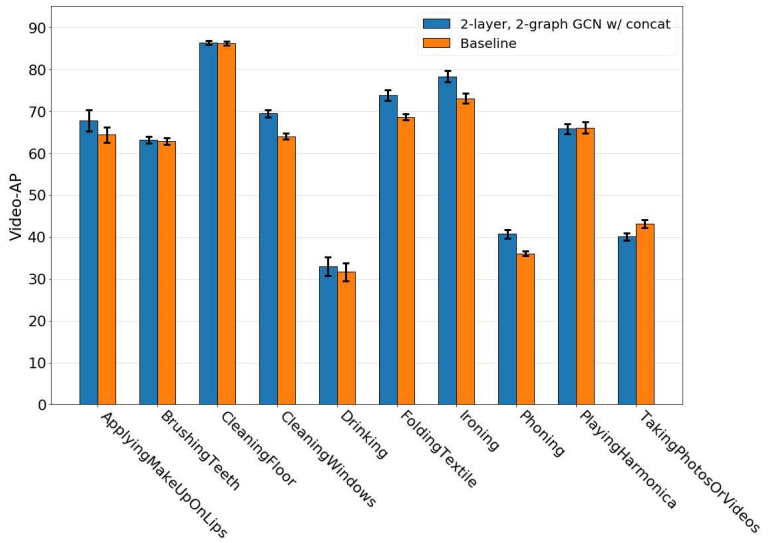

Figure 3: Average precision per class on the test set across five repetitions of GCN and baseline model.

corresponding increase is 2.24 points (3.75%) in mean mAP and 2.94 points (4.91%) in maximum mAP. On a significance level of 0.05, one-sided t-test and Mann-Whitney-Wilcoxon test provided significant evidence of a greater mean for the GCN model, with p-values 0.00016 and 0.0039, respectively. Figure 3 illustrates per-class average precision for the baseline and GCN model. GCN performs comparably or better than the baseline model in all classes except "TakingPhotosOrVideos".

Comparing our GCN model with the state-of-the-art [11, 55] on the test set, we obtain slightly improved performance in comparison to [55], while Chesneau et al. [11] achieve better performance by 1.69 points in mean mAP and 0.78 points in maximum mAP. As mentioned, however, our method is not directly comparable to [11, 55]. Finally, we note that our approach employs only RGB input, while [11, 55] use RGB and optical flow.

4.9 Reducing Annotation to One Bounding Box

Up to this point, all of our models have been trained with sparse spatial supervision using up to five annotated frames per action instance. For the following experiment, we examine the impact of the number of annotated frames on model's performance.

We label tubes as described in Section 3.4, but we restrict the number of annotated frames to one by randomly choosing a keyframe for each action instance. Then, we train a 2-layer, 2-graph GCN model using the newly labeled tubes.

During inference time, we evaluate our model on the test set in two ways: 1) we use ground truth

# Training GT Keyframes	# Evaluation GT Keyframes	Test mAP
1	1	58.67
1	up to 5	61.07
up to 5	up to 5	61.82 (± 0.51) (62.73)

Table 6: Test mAP with respect to different sparse supervision settings using one and up to five annotated frames for each action instance during training and evaluation.

tubes comprised of one randomly chosen annotated frame in order to evaluate the model in a realistic setting where the test set is annotated in the same way as the training set; 2) we report results using ground truth tubes comprised of up to five annotated frames in order to evaluate the effect on performance when training with minimum spatial supervision.

Results are shown in 6. Tubes in the training set are labeled using ground truth tubes comprised of a number of keyframes shown in the first column of Table 6. Accordingly, ground truth tubes in the test set consist of a number of annotated frames shown in the second column. Evaluating with up to five keyframes and training a model with one keyframe instead of up to five, results in a small decrease in test mAP from 61.82 to 61.07. Therefore, by annotating significantly less frames in the training set, we obtain a decrease in performance of less than one point in mAP. On the other hand, when we use one keyframe during evaluation, performance decreases by 2.4 points in mAP. The reason for such a decrease is that mAP is not estimated sufficiently well, since ground truth tubes consist of only one annotated frame.

We conclude that training a model with one keyframe per action instance and evaluating with up to five keyframes provides a good trade-off between annotation effort and model performance.

5 Analysis of Attention and Embeddings

In this section, we perform qualitative and quantitative analysis of our GCN model. In Section 5.1, we visualize the graphs' adjacency matrix in the form of attention maps, which highlight the context regions the model pays most attention to. In previous works [13, 45, 47], attention maps are evaluated only qualitatively. In this paper, we go one step further by quantitatively evaluating the ability of the model to highlight the relevant context. To this end, in Section 5.2, we introduce a metric to quantitatively evaluate the attention maps. The proposed metric is based on the recall of retrieved context, such as objects relevant to the action. In Section 5.3, we evaluate the ability of our model to pay attention to contextual cues in a zero-shot setting, i.e. for action classes and objects unseen during training. Finally, in Section 5.4, we visualize learned feature representations of actions and objects as t-SNE embeddings.

5.1 Visualization of Attention Maps

The adjacency matrix contains the relation or attention values, indicating the importance of every context node to the actor node. By visualizing the adjacency matrix, we obtain an attention map that highlights, for a given actor, the important context regions the model attends to. The highlighted regions comprise the context added to the actor via weighted averaging of context features.

For a detected actor, the adjacency matrix is calculated during the forward pass by computing the dot-product of the actor features with the context features. The adjacency matrix has $T' \times H' \times W' = 8 \times 14 \times 14$ entries, as context features comprise every 1×1 spatio-temporal location of the output feature map of `Mixed_4f` layer. The adjacency matrix is interpolated to the original frame size using bicubic interpolation over 4×4 neighborhoods.

Figure 4 illustrates a set of attention maps per class for a 2-layer, 2-graph GCN model with concatenation as merging function. Each attention map is the combination of four adjacency matrices

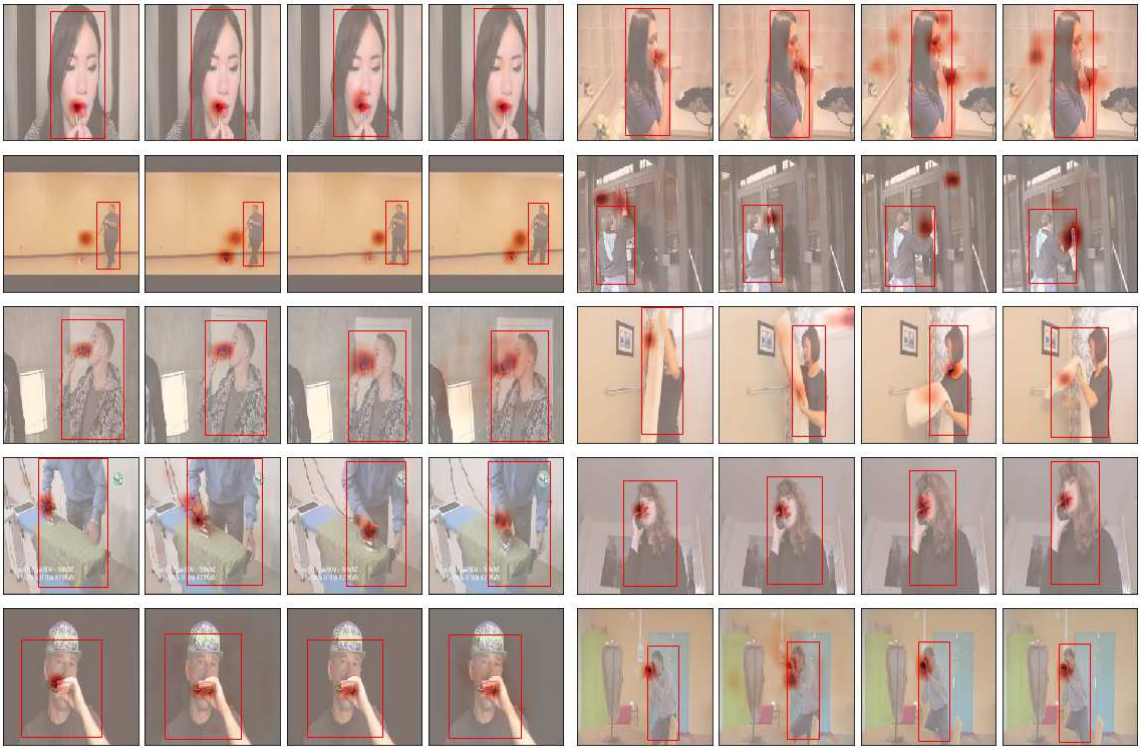

Figure 4: Learned adjacency matrix visualized as an attention map highlights the context regions considered most important by the model for recognizing the action. We provide one example per class. Horizontal axis represents time progression throughout the video clip. From left to right: "ApplyingMakeUpOnLips", "BrushingTeeth", "CleaningFloor", "CleaningWindows", "Drinking", "FoldingTextile", "Ironing", "Phoning", "PlayingHarmonica" and "TakingPhotosOrVideos".

(two graphs per layer) by summing their values along the spatial dimensions. Additionally, each example contains four attention maps along the x-axis, representing time progression along the input video clip.

The attention maps show that our GCN model attends to relevant context, such as objects, hands and faces. For example, the model focuses on the lips and pencil in "ApplyingMakeUpOnLips" action, on the toothbrush and hand in "BrushingTeeth" action and on the harmomica in "PlayingHarmonica" action. Furthermore, it is interesting that the model is able to track objects along time, for example, in "CleaningWindows" and "Ironing" actions.

We visualize success and failure cases in Figure 5. On the left-hand side of the first row, the attention misses the relevant object and focuses on the actor's hand instead. Nevertheless, the action "Ironing" is correctly classified. Although the attention misses the relevant object, it is possible that the visual appearance, motion and position of the hand still provide enough contextual cues. On the right-hand side, the model focuses on the object of interest, but the action is classified as "CleaningWindows" instaed of "TakingPhotosOrVideos". Further, on the left-hand side of the second row, the model adds context to the wrong actor resulting in classifying background as "CleaningFloor". Finally, in the last example, the attention map misses the relevant object and classifies the action as "PlayingHarmonica" instead of "Drinking".

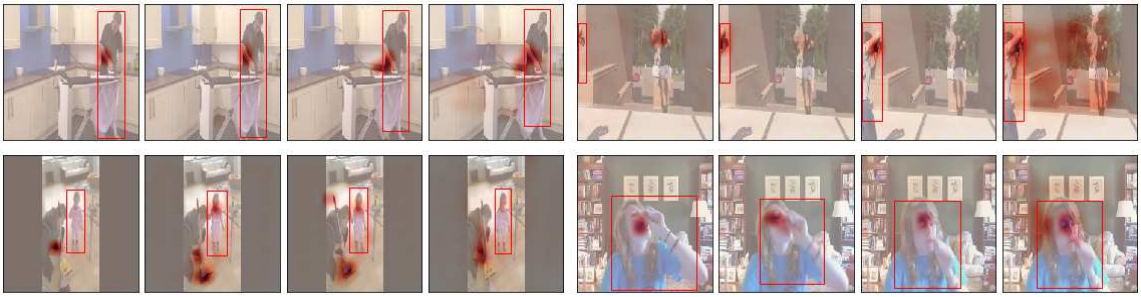

Figure 5: Success and failure cases of predictions and attention maps. On the top left example, the attention map misses the object, but the action is classified correctly. On the top right, although attention focuses on the relevant object, the action is misclassified. Bottom left, attention is added to the wrong actor. Finally, attention misses the object and the action is classified as "PlayingHarmonica" instead of "Drinking".

5.2 Quantitative Evaluation of Attention Maps

We introduce a metric to evaluate how well the model attends to relevant context. DALY consists of 10 human-object interaction actions, and therefore, objects comprise the most important contextual cue for recognizing actions. We expect our model to attend to relevant context so as to aid action classification, that is, we expect the attention maps to highlight objects relevant for recognizing the action.

In order to evaluate how well the attention maps highlight relevant objects, we introduce a metric based on recall of objects; the proportion of retrieved objects out of the total amount of objects, as highlighted or retrieved by attention maps. The location of objects, such as "phone" for action "Phoning", is known, since DALY provides object bounding box annotations on annotated frames, also called, keyframes.

Given the attention map produced for a detected actor on the keyframe, we sum the attention values inside the object's bounding box. An attention threshold varying from 0 to 1 defines correct instances, i.e. whether the object has been retrieved or not. An instance is a true positive if the sum of attention values is larger than the threshold. Otherwise, an instance is considered a false negative. The aforementioned process is summarized in a single curve by plotting the recall on the y-axis and the attention threshold on the x-axis. Note that the attention threshold varies from 0 to 1, since the attention map is a discrete probability distribution due to softmax normalization (Section 3.3.1).

Per-class recall curves are shown on left plot of Figure 6 as solid curves. For six out of ten action classes ("Ironing", "FoldingTextile", "Drinking", "PlayingHarmonica", "CleaningFloor", "BrushingTeeth"), the recall of retrieved objects remains high even even for large attention thresholds, with 40% to 60% of objects concentrating 100% of the attention. For actions "CleaningWindows", "ApplyingMakeUpOnLips" and "TakingPhotosOrVideos", almost 40% of objects contain 50% of the attention. Finally, "Phoning" class exhibits the lowest recall for thresholds greater than 15%.

As it is more likely for the attention to fall within a large object bounding box, we introduce the following normalized variant of the metric to account for object's box size. We multiply the added attention values inside the bounding box with a normalization coefficient, which is inversely proportional to object's bounding box size. The coefficient can be written as, $w_i = 1 - \frac{b_i}{f_i}$, $0 \leq w_i < 1$, where b_i is the area of the bounding box and f_i is the total area of the frame. In this way, we reduce the summed attention corresponding to large object bounding boxes and we promote the summed attention of small bounding boxes.

The right plot of Figure 6 illustrates results of normalized recall curves. Smaller recall is obtained for large values of attention threshold (i.e. 90%-100%) for almosts all classes. For the rest of the thresholds, the largest decrease in recall is obtained for classes "CleaningFloor" and "FoldingTextile",

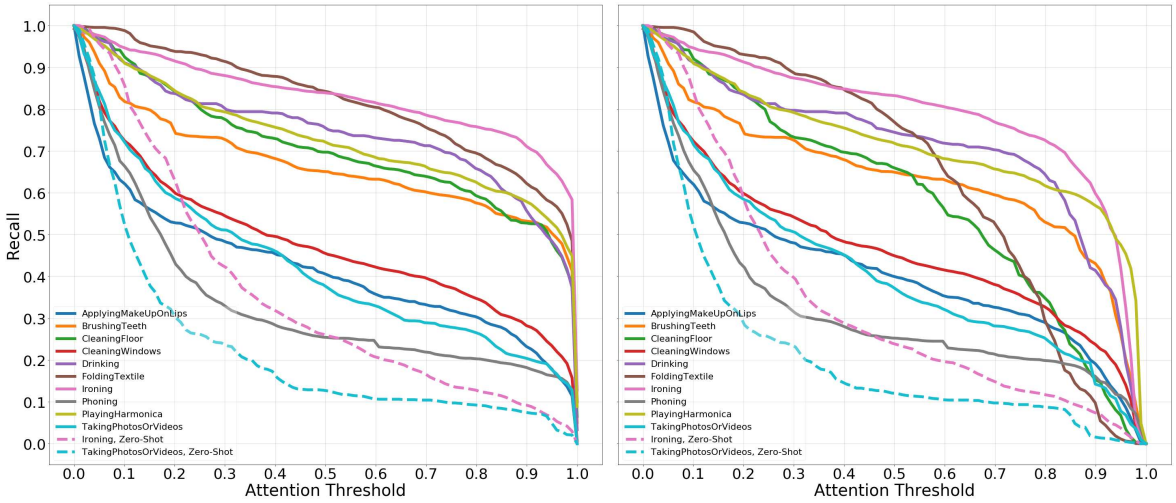

Figure 6: We evaluate how well the attention maps focus on relevant objects by plotting the recall of retrieved objects for a varying attention threshold. The plot on the right-hand side illustrates curves normalized with respect to object's bounding box size. Dashed curves evaluate the attention maps in a zero-shot setting for classes that the model has not been trained to recognize.

which usually contain relatively large objects, such as mop, towel and cloth.

5.3 Zero-Shot Attention Maps

In the following, we evaluate the attention maps on a zero-shot setting. Namely, we evaluate how well our model generalizes on modeling context for objects and action classes unseen during training. We expect the model to be able to relate an unseen action with its relevant context. Intuitively, the dot-product can be seen as a distance metric which measures the similarity or relation between actor features and context features. During training, the actor and context transformations learn a projection of these features so that a large dot-product corresponds to a large similarity or relation. Therefore, for an unseen action class, we expect the projected features of the actor to be more related, i.e. have a larger dot-product, to a relevant context feature, compared to irrelevant ones.

To this end, we train a 2-layer, 2-graph GCN model by excluding two action classes. In order to avoid overfitting, we perform a simple interpolation based on the total number of tubes in the training set in order to determine the total number of training epochs. Accordingly, we train the model for 344 epochs, instead of 450. During inference time, we extract the attention maps for input clips belonging to excluded classes. We evaluate the attention maps as described in Section 5.2.

Figure 7 illustrates success and failure cases of attention maps for two excluded classes, namely, "Ironing" and "TakingPhotosOrVideos". The first two rows present two examples per class, where the attention focuses on relevant objects even though the model has not been trained to recognize these actions, nor encode context with respect to the objects present. The last two rows display two examples per class where the model was unable to focus on relevant objects, and focuses mostly on the actor's head instead.

An evaluation of zero-shot attention maps for two excluded classes is shown with dashed curves in Figure 6. The right plot contains the normalized variant of the metric. Given the zero-shot setting, we obtain a relatively high recall for small attention thresholds, with approximately 40% and 25% of objects retrieved for "Ironing" and "TakingPhotosOrVideos", respectively, for 30% attention threshold. Although our model has not been trained to recognize the two actions, it is able to highlight their relevant objects.

Figure 7: Zero-shot attention maps for classes "Ironing" and "TakingPhotosOrVideos". Although the GCN model has not been trained to recognize these two actions nor their context, it is able to focus on relevant objects. The first two rows illustrate such cases. The last two rows show failure cases, where the attention usually focuses on the actor's head, instead of the object.

5.4 Embeddings of Actions and Objects

In this section, we visualize learned feature representations of actions and objects as produced by the GCN and baseline model. We embed these feature representations in a 2-dimensional space using t-distributed Stochastic Neighbor Embedding (t-SNE) [48].

For the GCN model, we visualize the updated actor features, i.e. the original actor features with added contextual cues as a result of applying graph convolutions (Section 3.3.2). Object features are obtained from the output of 3D convolutional layer, $\mathbf{W}_\phi$, by selecting the activations corresponding to the object bounding box and then applying spatial average pooling. Actor and object features are 512-dimensional as a result of concatenating the outputs of two graphs of the last GCN layer in a 2-layer, 2-graph GCN model. For the baseline model, we visualize the actor features produced by the I3D tail, which are 1024-dimensional. Object features, which are 832-dimensional, are obtained from the output feature map of `Mixed_4f` layer.

Embeddings for actions are shown in the top-left and lower-left of Figure 8 for the GCN model and baseline model, respectively. Compared the baseline model, the GCN model produces more distinct clusters with smaller intra-group variance. It is interesting that actions are separated in three super-clusters, which is more apparent for the GCN model. The first one is formed from "CleaningFloor" and "CleaningWindows" actions, the second consists of "Ironing" and "FoldingTextile" and the third contains "TakingPhotosOrVideos", "PlayingHarmonica", "Drinking", "Phoning", "BrushingTeeth", "ApplyingMakeUpOnLips". Indeed, actions within super-clusters are similar in terms of visual posture and motion.

Object embeddings are shown in the top-right and lower-right of Figure 8 for the GCN model and baseline model, respectively. Although the clusters produced by the two models are similar, the GCN model is still able to produce relatively tighter clusters. We observe that clusters of similar objects are located closer to each other, e.g. "camera" and "phone", "bedsheet" and "towel", (cleaning) "cloth", "scrubber" and "squeegee". A possible explanation is that similar objects share similar visual characteristics and/or are semantically similar i.e. relevant objects for recognizing the same

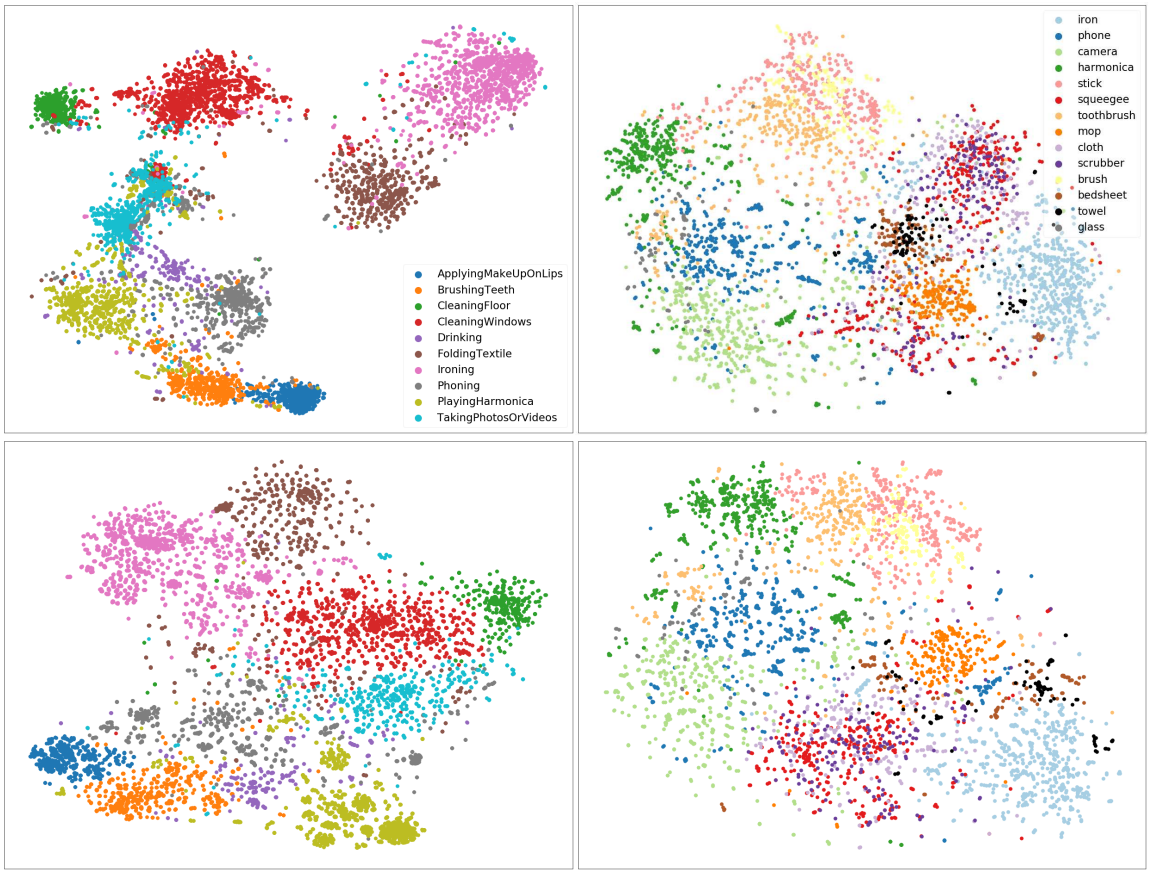

Figure 8: Left: T-SNE embeddings of actions and objects for a 2-layer, 2-graph GCN model (top) and baseline model (bottom). For actions, every point corresponds to a detected actor and color represents the action class. For objects, every point corresponds to a different object and color indicates the type of object.

action class.

6 Discussion

We discuss similarities and differences between our method and related work [13,45,47,60] on modeling context for action detection in Section 6.1. Further, we reflect on the results and analysis of Section 4 and Section 5 in Section 6.2 and Section 6.3, respectively. Finally, we discuss limitations of our approach and directions for future work in Section 6.4.

6.1 Comparison to Related Work

The relation network [40], initially proposed for visual question answering, is extended in [45, 47] to model context for action detection. Similarly to these works, we adopt the idea introduced in [40, 45] to treat every 1×1 location in the feature map as context. In [45, 47], learned relation features between the actor and the context are used for action classification. In contrast, we classify actions using contextualized actor features obtained by applying graph convolutions on a graph structure. In [45, 47], per-class attention maps are produced by relying on class activation maps [63], while our

model is able to highlight the exact learned context by visualizing the adjacency matrix as an attention map.

Girdhar et al. [60] extend the Transformer architecture [49] to model context for action detection. Similar to their work, we also use dot-product attention to model relations between the actor and the context, while a weighted average of context features produce updated actor features. In contrast to [60], we formulate contextual modeling as a graph learning problem, in which, relational reasoning is performed using Graph Convolutional Networks [26]. Although a Transformer can be represented as a graph neural network and vice versa, we argue that a graph representation is simpler and more intuitive compared to the Transformer representation of Queries, Keys and Values. Whilst [60] represent context using Keys and Values obtained by two linear transformations of the feature map, this mechanism does not have a direct interpretation in a graph. To this end, we use a single linear transformation of the feature map, called the context transformation, that outputs transformed context features. Context features correspond to nodes in a graph, called context nodes. Therefore, since we represent context using a single linear transformation of the feature map, we consider Keys to be equivalent to Values. Furthermore, our architecture uses no residual connections nor Layer Normalization (LayerNorm), two essential components of the Transformer. Although LayerNorm is irreplaceable in [49], it can increase overfitting [58] and its placement in the architecture can make the difference between stable and unstable training [7, 31]. Additionally, it is worth noting that Girdhar et al. [60] use scaled dot-product [49], while we employ its unscaled counterpart as it provided a significant increase in performance. Finally, although [60] produce similar attention maps, the attention maps produced by our architecture are obtained by visualizing the graph's adjacency matrix.

Similar to our work, Wu et al. [56] and Zhang et al. [60] employ Graph Convolutional Networks [26] to model contextual cues. Nodes in the graph represent detected actors [56] and detected actors and objects [60]. In contrast, we do not restrict our model to a specific set of actors and/or objects to accumulate context from, but rather, our model implicitly learns the most relevant context regions to pay the most attention to. Hence, our approach is suitable to reason with respect to arbitrary context and objects which cannot be detected, e.g. because the object detector has not been trained to do so. Finally, [56, 60] are not able to produce attention maps, as their adjacency matrix is restricted to represent actor and/or object detections.

All aforementioned approaches [13, 45, 47, 60] rely on full actor supervision during training. We show that learning context can be achieved using weak actor supervision, requiring up to five, or even one, actor bounding box annotation throughout the action instance. Furthermore, our work is the first to quantitatively evaluate how well the attention focuses on important context and evaluate model's generalization in a zero-shot setting.

6.2 Experimental Results

The experimental results show that learning contextual cues improves performance with respect to Video-mAP compared to a baseline model which does not use context. In Section 4.3, we show that using two GCN layers and constructing multiple graphs per layer is beneficial. Additionally, we find that concatenation outperforms summation for combining the output of multiple graphs.

In Section 4.5, we show that performance improves by adding spatial information in the graph with respect to actor and context coordinates. This result indicates that the model is able to learn spatial relations between the actor and the context.

We find that I3D tail plays an important role in encoding actor features, in Section 4.6. Removing I3D tail results in a decrease of more than four points in mAP. This result suggests that the rest of I3D backbone (prior to I3D tail), which encodes the whole input clip, fails to encode fine-grained actor features. Accordingly, fine-grained actor features are handled by the I3D tail, which is placed directly after RoI Pooling.

Results in Section 4.7 suggest that weight initialization affects model's performance. We propose to slightly reduce the range of uniform initialization for the context transformation in the first GCN

layer. We experimentally demonstrate that combining this adjustment with a proper choice of gain is beneficial in terms of performance.

With respect to weak supervision, in Section 4.9, we train our model with minimum spatial supervision, i.e. using only one annotated frame per action instance. Results suggest that performance is comparable to training a model using up to five annotated frames. Nevertheless, a requirement is that the model is evaluated during inference using up to five annotated frames so as to sufficiently estimate Video-mAP.

6.3 Attention

Section 5 presents a qualitative and quantitative analysis of our GCN model. The adjacency matrix, visualized in the form of attention maps, focuses on important context regions, such as objects, hands and faces. Additionally, the attention tracks important context regions across time. Our proposed metric, based on recall of retrieved objects, validates that the attention highlights objects relevant to the action. The GCN model is able to even attend to relevant objects in a zero-shot setting, namely, for actions unseen during training.

6.4 Future Work

In the following, we outline promising directions for future work. First, we discuss limitations of our model and propose research directions to overcome these limitations. Finally, we discuss future work with respect to extensions of our existing work.

One limitation is the requirement of an external detector to localize actors. As a result, our approach does not address detection and classification jointly. Therefore, an interesting direction could be the development of models for end-to-end weakly-supervised action detection. Furthermore, relative spatial information, instead of absolute coordinates, could be more effective in learning relations between the actor and the context. Another research direction is to obtain a better understanding of how weight initialization affects model's performance and develop effective initialization techniques.

With respect to extensions of our current work, including a second stream that operates on optical flow would be beneficial for model performance. Furthermore, we are interested in addressing relational reasoning on a hierarchical level. Specifically, our current model reasons with respect to context on a low level, aggregating context around the actor in a short temporal interval. Hierarchical reasoning could investigate reasoning on a higher level, such as among consecutive clips to model longer temporal dependencies and clips of different videos from the same or related action(s). Finally, as our model is not restricted to a particular attention mechanism, it would be interesting to examine other mechanisms for modeling relations, such as additive attention [2, 40, 50].

7 Conclusion

An approach is proposed to learn context, such as actor-actor and actor-object interactions, so as to improve action classification for the task of action detection. Our models learns spatio-temporal context for a detected actor by performing relational reasoning on a graph structure using Graph Convolutional Networks. We show that our model outperforms a baseline which uses no context. The learned adjacency matrix, visualized as an attention map, aids explainability by highlighting the learned context, such as objects, hands and faces. We evaluate how well the model learns relevant context using our proposed metric, which shows that objects relevant for recognizing the action are retrieved with high recall by the attention. Additionally, we demonstrate learning of context in a zero-shot setting: our model is able to learn what object is relevant for recognizing an action, even though it has not been trained to recognize the action. All the above are achieved in a weakly-supervised setting using only up to five or even one actor bounding box annotation per action instance.

References

[1] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, pages 3686–3693, 2014.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, Conference Track Proceedings*, 2015.

[3] N. Ballas, L. Yao, C. J. Pal, and A. C. Courville. Delving deeper into convolutional networks for learning video representations. *CoRR*, abs/1511.06432, 2015.

[4] F. Baradel, N. Neverova, C. Wolf, J. Mille, and G. Mori. Object level visual reasoning in videos. *CoRR*, abs/1806.06157, 2018.

[5] H. S. Behl, M. Sapienza, G. Singh, S. Saha, F. Cuzzolin, and P. H. S. Torr. Incremental tube construction for human action detection. *CoRR*, abs/1704.01358, 2017.

[6] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[7] M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. F. Foster, L. Jones, N. Parmar, M. Schuster, Z. Chen, Y. Wu, and M. Hughes. The best of both worlds: Combining recent advances in neural machine translation. *CoRR*, abs/1804.09849, 2018.

[8] Y. Chen, M. Rohrbach, Z. Yan, S. Yan, J. Feng, and Y. Kalantidis. Graph-based global reasoning networks. *CoRR*, abs/1811.12814, 2018.

[9] J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading. *CoRR*, abs/1601.06733, 2016.

[10] G. Chéron, J. Alayrac, I. Laptev, and C. Schmid. A flexible model for training action localization with varying levels of supervision. *CoRR*, abs/1806.11328, 2018.

[11] N. Chesneau, G. Rogez, K. Alahari, and C. Schmid. Detecting parts for action localization. *CoRR*, abs/1707.06005, 2017.

[12] V. Escorcia, C. D. Dao, M. Jain, B. Ghanem, and C. Snoek. Guess where? actor-supervision for spatiotemporal action localization. *CoRR*, abs/1804.01824, 2018.

[13] R. Girdhar, J. Carreira, C. Doersch, and A. Zisserman. Video action transformer network. *CoRR*, abs/1812.02707, 2018.

[14] R. Girdhar and D. Ramanan. Attentional pooling for action recognition. *CoRR*, abs/1711.01467, 2017.

[15] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[16] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[17] G. Gkioxari and J. Malik. Finding action tubes. *CoRR*, abs/1411.6031, 2014.

[18] C. Gu, C. Sun, S. Vijayanarasimhan, C. Pantofaru, D. A. Ross, G. Toderici, Y. Li, S. Ricco, R. Sukthankar, C. Schmid, and J. Malik. AVA: A video dataset of spatio-temporally localized atomic visual actions. *CoRR*, abs/1705.08421, 2017.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[21] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] R. Hou, C. Chen, and M. Shah. Tube convolutional neural network (T-CNN) for action detection in videos. *CoRR*, abs/1703.10664, 2017.

[23] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black. Towards understanding action recognition. In *International Conf. on Computer Vision (ICCV)*, pages 3192–3199, Dec. 2013.

[24] V. Kalogeiton, P. Weinzaepfel, V. Ferrari, and C. Schmid. Action tubelet detector for spatio-temporal action localization. *CoRR*, abs/1705.01861, 2017.

[25] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, June 2014.

[26] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[29] P. Mettes, C. G. M. Snoek, and S. Chang. Localizing actions from video labels and pseudo-annotations. *CoRR*, abs/1707.09143, 2017.

[30] P. Mettes, J. C. van Gemert, and C. G. M. Snoek. Spot on: Action localization from pointly-supervised proposals. *CoRR*, abs/1604.07602, 2016.

[31] T. Q. Nguyen and J. Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

[32] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. *CoRR*, abs/1606.01933, 2016.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[34] R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304, 2017.

[35] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *European conference on computer vision*, pages 744–759. Springer, 2016.

[36] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[38] S. Saha, G. Singh, and F. Cuzzolin. Amtnet: Action-micro-tube regression by end-to-end trainable deep architecture. *CoRR*, abs/1704.04952, 2017.

[39] S. Saha, G. Singh, M. Sapienza, P. H. S. Torr, and F. Cuzzolin. Deep learning for detecting multiple space-time action tubes in videos. *CoRR*, abs/1608.01529, 2016.

[40] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4967–4976. Curran Associates, Inc., 2017.

[41] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *CoRR*, abs/1506.04214, 2015.

[42] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[43] G. Singh, S. Saha, M. Sapienza, P. H. Torr, and F. Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3637–3646, 2017.

[44] P. Siva and T. Xiang. Weakly supervised action detection. In *BMVC*, 2011.

[45] C. Sun, A. Shrivastava, C. Vondrick, K. Murphy, R. Sukthankar, and C. Schmid. Actor-centric relation network. *CoRR*, abs/1807.10982, 2018.

[46] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[47] O. Ulutan, S. Rallapalli, M. Srivatsa, and B. S. Manjunath. Actor conditioned attention maps for video action detection. *CoRR*, abs/1812.11631, 2018.

[48] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[51] H. Wang and C. Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision*, Sydney, Australia, 2013.

[52] X. Wang, R. B. Girshick, A. Gupta, and K. He. Non-local neural networks. *CoRR*, abs/1711.07971, 2017.

[53] X. Wang and A. Gupta. Videos as space-time region graphs. *CoRR*, abs/1806.01810, 2018.

[54] P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Learning to track for spatio-temporal action localization. *CoRR*, abs/1506.01929, 2015.

[55] P. Weinzaepfel, X. Martin, and C. Schmid. Towards weakly-supervised action localization. *CoRR*, abs/1605.05197, 2016.

[56] J. Wu, L. Wang, L. Wang, J. Guo, and G. Wu. Learning actor relation graphs for group activity recognition. *CoRR*, abs/1904.10117, 2019.

[57] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 305–321, 2018.

[58] J. Xu, X. Sun, Z. Zhang, G. Zhao, and J. Lin. Understanding and improving layer normalization. pages 4381–4391. Curran Associates, Inc., 2019.

[59] Z. Yang, J. Gao, and R. Nevatia. Spatio-temporal action detection with cascade proposal and location anticipation. *CoRR*, abs/1708.00042, 2017.

[60] Y. Zhang, P. Tokmakov, C. Schmid, and M. Hebert. A structured model for action detection. *CoRR*, abs/1812.03544, 2018.

[61] J. Zhao and C. G. M. Snoek. Dance with flow: Two-in-one stream action detection. *CoRR*, abs/1904.00696, 2019.

[62] B. Zhou, A. Andonian, and A. Torralba. Temporal relational reasoning in videos. *CoRR*, abs/1711.08496, 2017.

[63] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *CoRR*, abs/1512.04150, 2015.