UNIVERSITÄT BONN

Modeling Human Actions in Multi-Label settings

Dissertation

zur

Erlangung des Doktorgrades (*Dr. rer. nat.*)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Sovan Biswas

aus

Barasat, Indien

Bonn 2025

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter/Betreuer: Prof. Dr. Jürgen Gall
Gutachterin: Prof. Dr. Hilde Kühne
Tag der Promotion: 09.07.2025
Erscheinungsjahr: 2025

Abstract

by Sovan Biswas

for the degree of

Doctor rerum naturalium

Human actions are often complex and occur in dynamic contexts, posing a challenge for traditional recognition models. This challenge is further exaggerated due to humans' innate multi-tasking nature, i.e. a person typically performs multiple actions at the same time. This thesis delves into multi-label human action recognition and analysis, bridging the gap between single and group activities. Furthermore, the thesis acknowledges the cost of labeled data required for training and discusses novel approaches to develop models in various fully supervised settings to data-scarce, weakly supervised environments.

The core contributions lie in developing novel neural network architectures that can capture the intricacies of multi-label action recognition. Our first contribution is on Structural Recurrent Neural Networks (SRNNs) for group activity analysis. These networks capture individual actions, interactions between individuals, and the overall group activity, agnostic to the size of the group. Moving from group activity, we also proposed a Hierarchical Graph-RNN that specifically tackles multiple individual actions. This architecture incorporates the temporal context and relationships between different actions to achieve accurate multi-label recognition in space and time.

Beyond fully supervised settings, we also explored weakly supervised learning, where action annotations are scarce. Here, our approaches rely on sets of actions instead of individual classes as annotations that are cost and time-effective to obtain. Our initial approach uses Multi-Instance Multi-Label (MIML) Learning followed by constraint-based Linear programming to map the set of actions to individual humans in a video. Furthermore, the thesis addresses the challenge of longer videos in weakly supervised settings. Here, a novel Multiple Instance Triplet Loss (MITL) exploits temporal similarity across consecutive frames in comparison to temporal distant frames to train the action recognition model effectively.

Through this dissertation, we advanced the state-of-the-art in multi-label action analysis, proposed novel architectures for group and individual action recognition exploiting temporal and spatial context, and finally, explored approaches to develop models for weakly supervised settings. We demonstrated the effectiveness of our approaches through comprehensive experimentation and by comparing them with existing state-of-the-art on well-known public

benchmarks. In the end, we conclude by discussing the open challenges and possible future research directions for multi-label human action analysis.

Keywords: multi-label action recognition, spatio-temporal action detection, weak-supervision.

Contents

List of Figures

List of Tables

List of Publications

This dissertation is based on the following publications:

- Sovan Biswas, and Jürgen Gall.
  "Structural Recurrent Neural Network (SRNN) for Group Activity Analysis"
  In IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2018.
  DOI: https://doi.org/10.1109/WACV.2018.00180

- Sovan Biswas, Yaser Souri and Jürgen Gall.
  "Hierarchical Graph-Rnns for Action Detection of Multiple Activities"
  In IEEE International Conference on Image Processing (ICIP), 2019.
  DOI: https://doi.org/10.1109/ICIP.2019.8803650

- Sovan Biswas, and Jürgen Gall.
  "Discovering Multi-label Actor-Action Association in a Weakly Supervised Setting"
  In Asian Conference on Computer Vision (ACCV), 2020.
  DOI: https://doi.org/10.1007/978-3-030-69541-5_33

- Sovan Biswas, and Jürgen Gall.
  "Multiple Instance Triplet Loss for Weakly Supervised Multi-Label Action Localisation of Interacting Persons"
  In IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), 2021.
  DOI: https://doi.org/10.1109/ICCVW54120.2021.00245

Chapter 1

Introduction

Contents

We have all heard a famous quote, "A picture is worth a thousand words". This phrase highlights the depth of information a single image can convey. A picture captures basic details, like colors and objects, at an elementary level. Nevertheless, in conjunction with other characteristics, this elementary information may convey higher-level details, such as the scene or the background location that a picture contains, for example, indoor or outdoor etc.

A video is a series of moving images. Thus, if an image is equivalent to a thousand words, a video is a million words due to the additional information captured by the changes in the series of images. This additional ability of videos to capture changes enables humans to utilize videos in many applications. Further, the massive technological developments in the ease of making, storing, and transmitting a video have made them ubiquitous. One can find them from short-duration clips captured on apps such as TikTok or Instagram to long-duration movies that convey a story or a movie, from the fixed viewpoint of surveillance cameras to live transmission of a football match. It is challenging to imagine a current or future world without videos.

On Youtube, a widely known site dedicated to storing and distributing video content, it has been reported that more than 500+ hours of video content are uploaded there every minute[1]. This massive quantity of videos needs to be analyzed for their content to perform meta-information generation, catalog videos in any storage system, recognize and filter violent content/illegal content, or retrieve specific content retrieval. This analysing task at this enormous scale is outside the scope of human capabilities and thus, the need for developing technologies for automated video analysis. Automated video analysis is also crucial to diverse applications such as surveillance systems, traffic analysis, AI assistance in human activity, etc. where high speed of processing and low margin of error are of utmost importance.

[1] https://blog.youtube/press/

Figure 1.1: Few varieties of videos capturing human actions: a) An example of simple action of *Walking* based on distinct motion pattern. (Schuldt et al., 2004) b) An example of complex activity of *Making Sandwich*. (Zhou et al., 2018) c) Complex group action (e.g. *Ice hockey*) with multiple persons involved. (Carreira and Zisserman, 2017)

Typically, most of the captured videos consist of humans as the protagonists. Thus, in this thesis, we analyze the content of the video through the lens of human behavior and actions.

1.1 Motivation

Any short-time distinct motion pattern that a human performs in the video is an *action*. An action can range from a short atomic pattern, such as *standing* or *walking* (Figure 1.1a), to a very complex one, such as gymnastics. Sometimes, a temporal sequence of such actions can form a complex *activity* (Figure 1.1b e.g. *making sandwich*). Further, actions can also involve single person or multiple persons as seen in *playing ice hockey* (Figure 1.1c). Identifying a wide variety of actions or activities can help in many applications. Traditionally, the significant focus of recent research is limited to such atomic actions or complex activities, ignoring a wide variety of other challenges with understanding the actions or activity. One of the guiding factors of the thesis is to explore human group activity analysis and multi-label action recognition/detection, unexplored research problems within action and activity understanding.

In many naturalistic settings, humans are in a group and interact with each other. This interaction could be synchronized, such as *Dancing*, or unsynchronised, such as *walking in a crowd*. In both cases, the action of one person in a group is highly contextualized based on the action of another person in the same group. For example, if a person in a group is *speaking* the other people in the group are *listening*. Apart

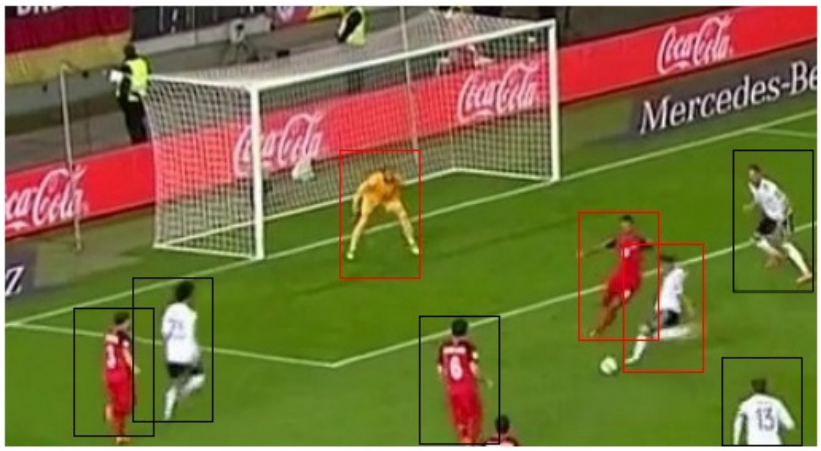


Figure 1.2: Example of *key action based group behavior*: The moment of a shooting in a soccer match, just before the goal. In this, the group action of "goal" is defined by the key actors of the striker, defender and the goalkeeper (marked by bounding boxes in Red). (Jiang et al., 2020)

from the neighbor context, people behave in tandem with others or individually to display a group behavior or activity. Understanding such nuances of individuals or groups is critical in various applications. For example, understanding and analyzing the behavior of each football player or the whole team in a sports video analysis can improve the player's and the team's performance. Alternatively, analysing each individual's behavior within a crowd can help in security or preemptive/quick nabbing of a rogue actor or crowd management.

Furthermore, humans are multi-taskers in nature. They perform multiple actions in a single time. For example, a person can be sitting and reading a book simultaneously, or a person can be walking and talking simultaneously. It is preposterous to have an automatic system to recognize only a single action of either *sit* or *read* at any given instant, resulting in incomplete information capture. This detailed understanding of nuances of human actions and their co-occurrence is vital for a holistic understanding of the content of the video or to generate grounded textual scene descriptions or summaries. In applications such as AI assistance to humans in their daily activities, understanding multi-label actions is crucial to provide holistic feedback to humans.

The thesis aims to explore new and feasible approaches to identify human actions in group and multi-label settings for a diverse range of real-world applications.

1.2 Challenges

Human actions are highly contextual. Typically, the context is temporal as exploited by temporal models (Richard et al., 2018; Kühne et al., 2018). In other words, there exist temporal relationships over time. However, actions could also be due to certain spatial situations. In this case, the actions depend upon the spatial context, such as object or persons and their actions. This interdependence on objects or (other) people is apparent in group activities like football, volleyball, or dancing, where the action of one person can directly influence the actions of others. It is therefore essential to distinguish each individual's actions and understand their impact on the group as a whole. Group behavior can be synchronized or unsynchronized. Synchronized group behavior, such as dancing, is when each actor behaves in a coordinated manner. On the other hand, humans in crowds are not synchronized due to their individuality and independence. The challenge for the group and individual action analysis lies in modeling the dependencies between actors despite this difference in coordination. Group behavior can also be considered of two types: a) key action-based group behavior or b) generalized group behavior. This is shown in Figure 1.2. A key action-based group behavior, especially in sports, is highly dependent on the key actions of a few players. The rest of the actors and their actions are generic/regular and occur in almost all other categories of group behavior. Thus, identifying the key actors and their actions is essential to identifying the group behavior and vice versa. For example, the group action of "goal" in football depends on identifying the striker's *shoot* action and the defender's *block* action. Other players are just running or walking, which is not distinctive. On the other hand, within generalized group behavior such as crowd, multiple people perform the same one or two individual actions and lack any distinctive action. This is a significant challenge as group characteristics are determined by different combinations of individual actions and their interactions between one or two actors within the group.

The basic assumption of humans performing a single action at a time is restrictive. Humans are multi-taskers. This is shown in Figure 1.3. However, identifying the open set of actions humans perform is difficult, as humans can perform one or multiple actions at a time. Specific actions such *listening* or *speaking* are ubiquitous and can occur in various scenarios such as indoor or outdoor. These background variations result in fewer visual cues to identify the actions with high accuracy. Additionally, this ubiquitous occurrence of some actions more than others results in a high class-imbalance. Despite these limitations, one can assume it is improbable that a person would be listening and speaking simultaneously at the same time. However, these actions are highly dependent on past temporal information and spatial information of other humans. In other words, a temporal dependency of action, for example, is when a person was speaking in recent past frames, then it is probable the person will continue speaking in the current and near-future frames, too. The spatial dependency is when a person speaks; it is evident that other persons in the

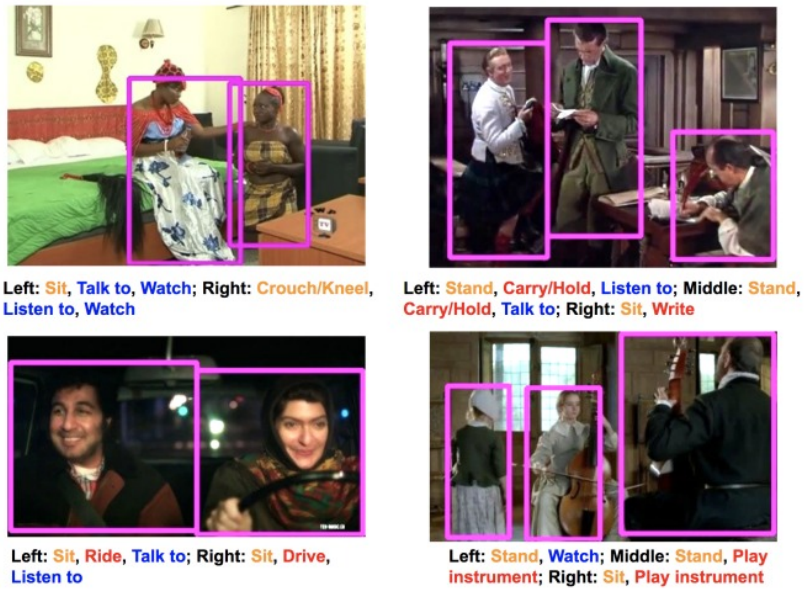


Figure 1.3: Example of Multi-Label Actions by human. Note, each actor performs variable number of actions. (Gu et al., 2018)

scene are listening. These minor details can help to identify the actions despite class imbalances, occlusions, and background variations and thus need spatio-temporal relationships.

Building a model with such spatio-temporal relations needs a considerable amount of annotated data. This is primarily a challenge regarding the cost and time needed to annotate a scene with multiple humans performing multiple actions. This limits the scalability of training data needed for effective real-world models. Reliability of annotations in such cases is another challenge as it requires significant focus and patience from the annotator. It is far easier to capture scalable and extensive training data with fewer details such as list of all actions in a video. Thus, in similar budget for cost and time allocated for annotation, one can have significantly more training data samples with fewer details as opposed to few detailed, reliable and accurate training data. Despite the huge amount of data in these cases, training a model from such partial and obscure annotations is a challenge.

1.3 Contributions

People can casually and effortlessly understand complex multi-label actions and group activities, making the task look easy. However, such an essential human capability has many layers of high-level semantical meanings. Thus, it is difficult for

computers to comprehend this task accurately despite recent research progress. As described in Section 1.2, understanding human group activities poses several difficulties, that range from group dynamics, class imbalances, open-set variations, annotations, etc. We now present our contributions in thesis that tackle the challenges of multi-label and group activity understanding tasks.

- In our first work presented in Chapter 4, we present a solution for the problem of group activity recognition. As explained before, the behavior of a group is defined by the interactions and behaviors of various individuals within the group. The proposed solution is focused towards joint (spatial and temporal) capturing of those interactions to identify the action of each person and the group.

- In Chapter 5, we present a solution for multi-label action recognition in group settings. In this research problem, the training data consists of the location of the humans and multiple the action labels for each humans. The proposed fully-supervised solution mainly focuses on capturing interactions between humans that aids in mutli-label action recognition.

- Chapter 6 proposes a semi-supervised solution for the multi-label action recognition and detection similar to Chapter 5. It is well known that annotating multiple labels for every human in a long video can be very slow and cost-intensive. Thus, this chapter presents a viable solution to learn multi-label action recognition and detection using semi-supervision.

- Chapter 7 proposes another semi-supervised solution for multi-label action recognition and detection overcoming the speed bottleneck of the earlier approach (Chapter 6). The approach uses a contrastive learning paradigm as opposed to discovering the pseudo-labels for each person through iterations, making the approach faster and less computationally expensive.

Chapter 2

Related Work

Contents

Now we review the key past work related to the research topic of the thesis. We start with our discussion and insights of the past works on group activity recognition. This is followed by discussions on the past works related to spatio-temporal and multiple activity detections. In the end, we briefly discuss some of the works on contrastive learning and multi-instance and multi-label learning as Chapters 6 and Chapters 7 heavily use these concepts.

2.1 Group Activity Recognition

As early in 2009, researchers (Choi et al., 2009) started analyzing the activities of a group or crowd. The authors (Choi et al., 2009) introduce crowd context in recognizing the activity being performed by each individual in the group. Traditionally graphical models with key contextual features (Choi et al., 2011; Choi and Savarese, 2014) have been deployed rigorously towards group analysis. However these models with handcrafted features (Ryoo and Aggarwal, 2009) of person tracks and movements, were outperformed by newer deep neural network architectures such as (Deng et al., 2016; Alahi et al., 2016; Shu et al., 2017). For group activity recognition, most of these deep neural networks are inspired by (Ibrahim et al., 2016a) which uses a multi-level cascade of recurrent neural networks for group activity recognition. In this approach, humans are detected and tracked to form multi-person tracklets. These tracklets along with their deep visual features are fed to the lower-level RNNs. The focus of these lower-level RNNs is to understand and model the actions of the individual persons. The higher level RNNs in the architecture instead focus on understanding the group activity. The individual actions and group activity predictions are done using Softmax in a feed-forward way. However, each method tackles a very different problem in the same framework. In (Shu et al., 2017), the

authors use a similar hierarchical architecture but the approach differs from previous work by proposing an energy-based approach that works significantly better if the amount of data is small. Furthermore, this approach also explores human interaction, but holistically by convolutional features extracted from both humans. In (Bagautdinov et al., 2017), the authors propose a joint approach for detecting humans and predicting their actions. Recently, approaches based on attention have been explored in (Yuan et al., 2021; Li et al., 2021). Due to the attention mechanisms, transformer-based architectures (Tamura et al., 2022; Zhou et al., 2022) have performed significantly better in recent past. However, these methods also require large amount of data and large attention computation memory to train effectively. To overcome the data requirement, variations of self-supervised contrastive learning have been used in (Chappa et al., 2023) to improve the performance. Recently, in (Wen et al., 2024) the authors explore the skeleton based features for multi-label group activity understanding.

Spatio-temporal graphs have been used in computer vision for various applications such as predicting human movements (Ionescu et al., 2014) or learning human activities and object affordances (Koppula et al., 2013). The spatio-temporal graphs presented in these works spatio-temporal relations between joints or joints and objects in a video. The methods (Koppula and Saxena, 2016; Amer et al., 2014) use handcrafted features along with graphical models, conditional random fields, or random forests for the aforementioned applications. Recently, neural networks have been deployed to solve spatio-temporal graph problems. For example, in (Deng et al., 2015), the authors use deep networks followed by inference using a probabilistic graphical model to recognize the actions in a group. The graph formulation with spatial and temporal relations was even used and extended in implicit transformer blocks of Groupformer (Li et al., 2021). Our approach, in Chapter 4, builds on the work (Jain et al., 2016) where a set of coupled RNNs are used to represent spatio-temporal graphs.

2.2 Spatio-Temporal Action Detection and Recognition

A common approach for action recognition and localization (Gkioxari and Malik, 2015; Hou et al., 2017; Kalogeiton et al., 2017; Singh et al., 2017) comprises the detection of the bounding boxes in each frame using object detectors (Girshick, 2015; Chao et al., 2018). The detected bounding boxes are then linked to obtain action tubes, which are then classified. These approaches, however, assume that every frame is annotated. Since such dense annotations are very time-consuming, many approaches (Weinzaepfel et al., 2016; Girdhar et al., 2018; Li et al., 2018) deal with sparse annotations where the action labels and locations are annotated only for a subset of frames, e.g., each frame per second. These works, however, treat each person independently despite in reality persons tend to interact with each other.

In the context of group activity analysis (Ibrahim et al., 2016a; Tang et al., 2018), the relations between various individual persons are used to infer the action label of the group as well as the individuals. The works (Ibrahim et al., 2016a; Tang et al., 2018) propose hierarchical models where the individual actions are modeled at the lower level and the group activity at the top level. While (Tang et al., 2018; Li et al., 2021; Tamura et al., 2022) assume that each individual is part of a sports team or group and each individual performs only one action as part of a group, (Gu et al., 2018) contains multiple activities per individual and only a subset of the actions are based on interactions with other individuals. Past works (Feichtenhofer et al., 2019; Fan et al., 2021), have shown improving underlying feature representation improves the multi-label action recognition even in group settings. Building upon stronger features, in (Pan et al., 2021; Rajasegaran et al., 2023), the authors further explored attention over human to human interactions within a group to improve multi-label spatio-temporal action detection. Recently, in (Gritsenko et al., 2024), the authors explored the advancements in detection transformers (DETR) (Carion et al., 2020) to form action tubes that are later used to perform multi-label action recognition.

There are several types of networks that can be applied to graphs like Graph Convolutional Networks (GCN) (Kipf and Welling, 2017; Duvenaud et al., 2015) or Graph Recurrent Neural Networks (Graph RNN) (Scarselli et al., 2009; Li et al., 2016; Jain et al., 2016). These graph networks have been used in various computer vision applications such as object detection (Qi et al., 2018), semantic segmentation (Qi et al., 2017), or visual question answering (Teney et al., 2017). For instance, in (Yang et al., 2018) the authors use an attentional GCN to model spatial relations between objects in an image. Recently, graph based audio-video understanding (Min et al., 2022; Min, 2023) was explored for ego-centeric videos and active-speaker detection over a very long temporal windows. In our work, in Chapter 5, we use a Graph RNN combined with a Scene RNN to model relations between different persons as well as temporal context.

2.2.1 Actor-Action Associations

Actor-action associations have been key to identify actions both in fully supervised and weakly supervised settings. In (Ghadiyaram et al., 2019), the authors perform soft actor-action association using tags as pre-training on a very large dataset for fully supervised action recognition. With respect to weak supervision, a few approaches use movie subtitles (Bojanowski et al., 2013; Laptev et al., 2008) or transcripts (Kühne et al., 2018; Richard et al., 2018) to temporally align actions to frames. In terms of actor-action associations for multiple persons, few approaches, such as (Ramanathan et al., 2016; Li et al., 2020), associate a single action to various persons. Recently, the authors of (Ulutan et al., 2020) explored actor associated action feature maps for implicit actor-action association as an alternative to region of Interest (RoIs). To the best of our knowledge, our work (Chapter 6) is the first

to perform explicit multi-person and multi-label associations.

2.3 Contrastive Learning

Contrastive learning, that maximises the intra-class similarity and minimises the inter-class similarity, has been used extensively over the years in various computer vision applications such as image representation learning (Chen et al., 2020; He et al., 2020; Oord et al., 2018), video representation learning (Qian et al., 2021; Behrmann et al., 2021), face recognition (Schroff et al., 2015; Deng et al., 2019), image captioning (Liu et al., 2018), phase grounding (Gupta et al., 2020), multi-view action recognition (Shah et al., 2023) or future prediction (Wu et al., 2020). Even though the objective of contrastive learning largely remained the same, the diverse formulations for intra-class and inter-class similarity has lead to various distinct loss functions such as the triplet loss (Hoffer and Ailon, 2015; Schroff et al., 2015), lifted-structure loss (Oh Song et al., 2016), N-pair loss (Sohn, 2016), angular loss (Wang et al., 2017), margin-based loss (Wu et al., 2017), multi-similarity loss (Wang et al., 2019), circle loss (Sun et al., 2020) and infoNCE loss (Oord et al., 2018). Our approach, in Chapter 7, proposes a loss that is based on the triplet loss (Schroff et al., 2015) but extends it to multiple instance learning in order to deal with the task of weakly supervised multi-label action localization.

2.4 Multi-instance Multi-label learning

In the past, many MIML algorithms (Nguyen et al., 2013; Nguyen, 2010) have been proposed. For example, (Zhou et al., 2012) propose the MIMLBoost and MIMLSVM algorithms based on boosting or SVMs. (Briggs et al., 2012) optimize a regularized rank-loss objective. MIML has been also used for different computer vision applications such as scene classification (Zhou and Zhang, 2006), multi-object recognition (Yang et al., 2017), and image tagging (Zha et al., 2008). (Lai et al., 2023) explored inter-label correlations and inter-instance correlations within the MIML framework for medical image classification. Recently, MIML based approaches have been used for action recognition (Li et al., 2020; Zhang et al., 2020).

Chapter 3

Preliminaries

Contents

In this chapter, we present an overview of concepts that are discussed throughout this thesis. The chapter starts with an introduction to neural networks, convolutional neural networks, and recurrent neural networks. This is followed by discussion on various video representation that is used for human activity analysis. We then briefly describe the localization task used for person identification. In the end, we discuss linear programming and techniques that are essential to understanding optimization for multi-label weak supervision.

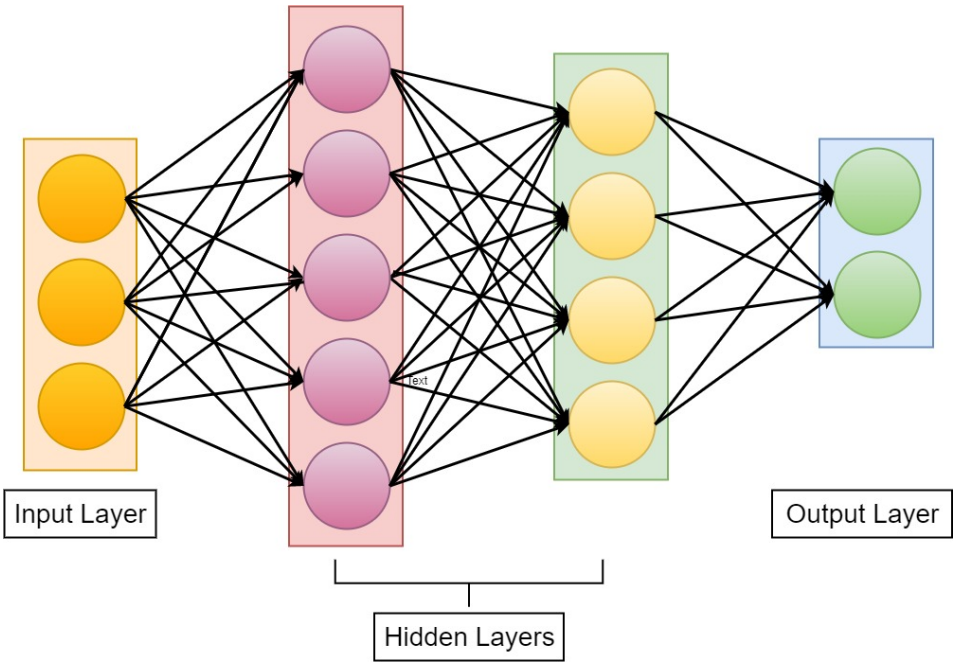


Figure 3.1: A standard feed-forward network with an input layer, two hidden layers and an output layer

3.1 Artificial Neural Networks

A biological neural network (McCulloch and Pitts, 1943) is a group of interconnected biological neurons to carry out a specific function when activated. Each neuron in the network is connected by a synapse. Motivated from the biology, an artificial neural network (Rosenblatt, 1958; Rumelhart et al., 1986) is defined as a group of artificial neurons called layers, where each layer receives information from the layer before. An artificial neuron is a processing unit that combines and applies transformations to the information from the previous layer. A standard layer of an artificial neural network can be divided into three categories: an input layer, hidden layers, and an output layer as shown in Figure 3.1.

The first layer is known as the input layer that predominantly intakes the data to be processed by the neural network. The last layer is known as output layer that generates the desired output post the processing of the input data. The hidden layers are the one or more layers that are sandwiched between the input and output layers. A standard neural network also known as feed-forward neural network or multi-layer perceptron (MLP) (Rumelhart et al., 1986) consists of one or more hidden layers. Each neuron, also known as node, is connected with edges to form a directed acyclic graph. Mathematically, the neurons are matrices, also known as parameters of the neural network, whose values are computed to achieve desired objective.

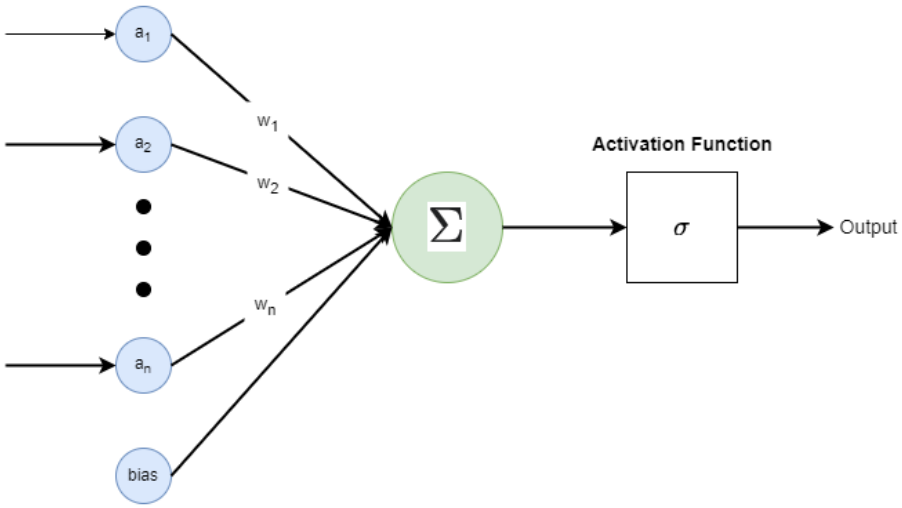


Figure 3.2: A simple artificial neuron with n inputs ($a_1, \ldots, a_n$), weights ($w_1, \ldots, w_n$), a bias and activation function σ.

In recent times, the neural networks are being widely used in the field of pattern recognition due to advancement of hardware computations and the ability of neural networks to approximate any function (Hornik et al., 1989). We, now, briefly describe the different key components of neural networks.

3.1.1 Artificial Neuron

An artificial neuron or simply **Neuron** is the building block of neural networks. It can be viewed as a system that takes inputs, applies a transformation on them, and produces an output which is called activation. A neuron is shown in Figure 3.2

A basic neuron computes a weighted sum of the input values, adds bias and then applies an activation function to the sum. The weighted sum and bias addition is a linear transformation function. However, the activation function, which is differentiable, is always a non-linear function. This non-linearity is crucial to create multi-layer neural network, otherwise the combination of all linear transformations at different layers is mathematically equal to a single linear transformation. A simple neural network is formed by stacking layers of neurons as shown in Figure 3.1. The weights and bias are learned based on the input data and desired output, through backpropagation as explained in Section 3.1.3. Mathematically, a neuron is explained

as follows:

$$o_i^l = \sum_{j=0}^{N} w_{ij}^l a_j^{l-1} + b_i^l \tag{3.1}$$

$$a_i^l = \sigma(o_i^l) \tag{3.2}$$

Here, the equations show the i^{th} output for neuron a_i at layer l. a_j^{l-1} is the j^{th} output from the previous layer $l-1$. $a^{l-1} \in \mathcal{R}^N$. (3.1) is the linear operation of the neuron and σ (3.2) is the non-linear activation function applied on the sum. w_{ij}^l and b_i^l are weights and bias, respectively. They both together are also known as learned parameters (θ_i^l) of the neuron. Some of the commonly used activation functions are Sigmoid, Hyperbolic Tangent, Relu, Softmax, etc.

The goal of a neural network is to map a specific input to a desired output. The desired output is generated by the output layer. As such, the non-linear function of the output layer is sometimes different and specifically designed to generate the desired output. For example, Sigmoid function is used for binary classification or Softmax function is used for multi-class classification.

3.1.2 Loss functions

A loss function computes the difference between the predicted output of the neural network and the desired output. This helps in evaluating how well the algorithm models the desired data. If the predictions are incorrect, the difference by the loss function is high. On the other hand, if the predictions are correct, the difference is low. Mathematically, a loss function is defined as:

$$L(x, y; \theta) = L(f_\theta(x), y) = L(\tilde{y}, y) \tag{3.3}$$

Here, (x,y) is a tuple of data. f_θ is the neural network that maps the input x to desired output y. θ are all the learnable parameters of the neural network, i.e. weights and biases of each neurons at different layers. $\tilde{y}$ is the predicted output of the neural network. A loss function L is a measure of the difference between the prediction and the desired output.

Typically, a loss function can be any function that measures the difference. However, for neural network it is important that the loss functions are differentiable as shown in Section 3.1.3. Depending upon the task and objective, various loss functions are popular such as *Mean Squared Error* and *Cross Entropy loss*.

3.1.3 Parameter Estimation

A neural network has multiple neurons at various layers and each neuron has parameters (θ). Typically, gradient descent is used as an optimization algorithm to

minimize the value of the loss function with respect to a given set of tuples of data (x,y) and initial value of the parameters. The optimisation is done by computing the change in the parameters θ iteratively for each tuple of data (x,y) by following the opposite direction of the gradients ($\delta L/\delta\theta$). This process is known as *stochastic gradient descent* (Robbins and Monro, 1951; Kiefer and Wolfowitz, 1952).

$$\theta_{k+1} = \theta_k - \eta\frac{\delta L}{\delta\theta_k} \tag{3.4}$$

Here, θ_k are the parameters of the network after the iteration k. η is the learning rate that controls the updation of the parameters in the direction of gradient. The learning rate can impact the performance of the model as large value might result in oscillation around the minima whereas a low value can result in long training time. In practice, batch gradient is used instead of single tuple of data to minimize the errors by average gradient approximation over a batch.

A typical neural network consists of multiple layers and large number of parameters. Any update of parameters at any layer impacts the values in the subsequent layers. Instead of computing the gradient for each weight individually, that can be intractable, Backpropagation (Rumelhart et al., 1986; LeCun et al., 1998) is used. This is done by computing gradient $\delta L/\delta\theta_i^l$ for each parameter θ_i^l at i^{th} position and layer l by using chain rule. This chain rule computation is performed using dynamic programming by iterating the computation one layer at time from last layer to avoid redundant calculations of intermediate terms.

3.2 Convolution Neural Network

Convolutional Neural Networks (CNNs) are the widely utilized type of artificial neural network in computer vision currently. It came into prominence with LeNet-5 (LeCun et al., 1998) where the authors used it for first time to recognize images of handwritten digits. Later, Alexnet (Krizhevsky et al., 2012) and VGG (Simonyan and Zisserman, 2014b) showcased its capability to understand real-word natural images. In videos, C3D (Ji et al., 2013) and I3D (Carreira and Zisserman, 2017) are popular architectures to capture the video's 3-D (spatial and temporal) information.

A standard multi-layer perceptron (MLP), where each neuron in a layer is connected to all the neurons of the previous layers, uses matrix multiplication in all the layers. However, in comparison, CNN uses convolution in atleast one of the layers. The use of convolution enables the network to exploit the information from spatial or temporal or other types of structural coherence. A standard convolution network consists of convolution layer, pooling layers and fully connected layers as seen in the architecture of LeNet-5 (LeCun et al., 1998) (Figure 3.3). We now describe briefly the two key types of layers used in CNNs.

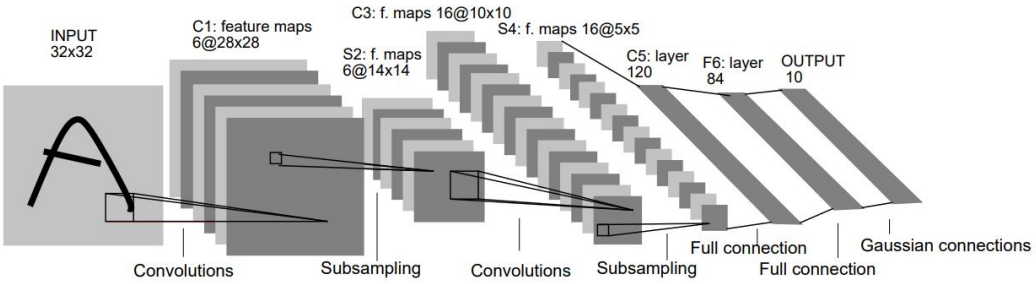


Figure 3.3: The network architecture of LeNet-5 (LeCun et al., 1998)

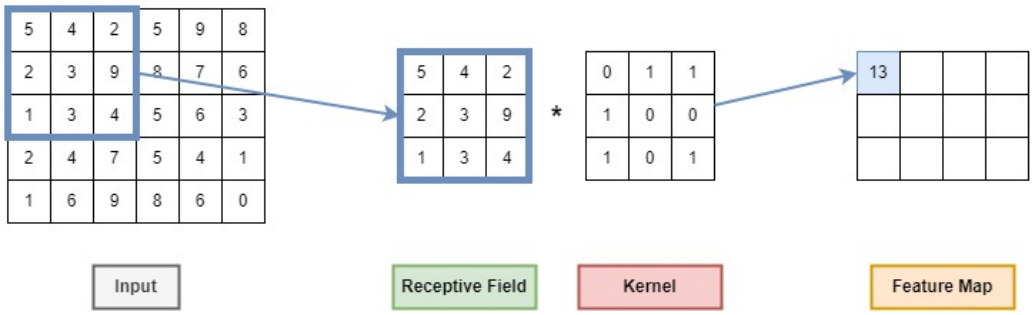


Figure 3.4: Functioning of a convolution layer

3.2.1 Convolutional Layer

Convolutional layers convolve the input from the previous layer and pass its result to the next layer, similar to any standard neural network. However, each convolution operation in a layer is defined by a kernel that shares parameters. They only process data within their receptive field. A kernel slides over the whole input in a fixed manner to generate a feature map. Due to the shared parameters within a kernel, CNNs have significantly less number of parameters in comparison to standard ANNs. Typically, 2D kernels are used to exploit spatial coherence, whereas 3D kernels are used for spatiotemporal coherence. Figure 3.4 shows the functioning of a convolutional layer.

3.2.2 Pooling Layer

A pooling layer is utilized to downsize the size of the feature map. This is done by combining the localized outputs of the previous layers into a single output. Two types of pooling layers are widely used a) Max pooling, where the output is the maximum value of each local region, or b) Average pooling, where the output is the average

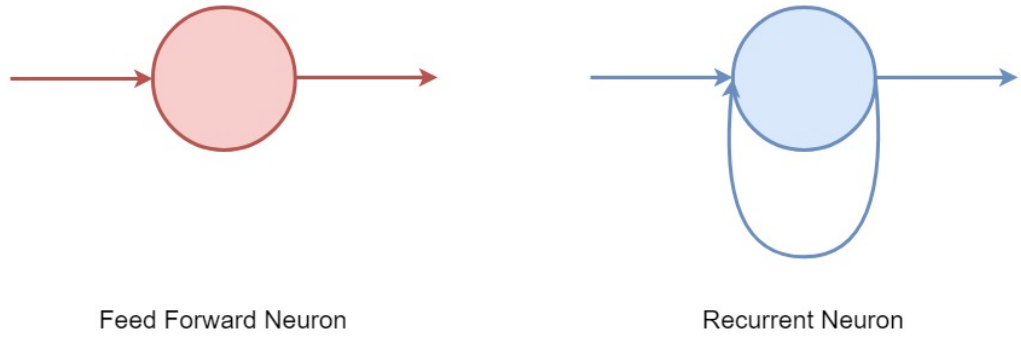


Figure 3.5: Comparison of feedforward neuron to recurrent neuron. In a recurrent neuron, the output of the neuron is feedback additionally as input.

value of each local region. Due to this, the pooling layer provides translational invariance that focuses on feature detection instead of their location.

3.3 Recurrent Neural Network

A recurrent neural network (RNN) is a type of artificial neural network that uses recurrent neurons instead of the standard feed-forward neurons. A recurrent neuron (also known as a cell) has feedback loops that connect the output back to its input. Further, RNNs have a hidden state that acts as a memory of the neuron. As such, this memory and recurrent neuron suits RNNs to process sequential or temporal, or serial data, where the value at the current time is related to the values in the past. RNNs have been very popular in applications with temporal problems, such as natural language processing (NLP), speech recognition, and image captioning. Figure 3.5 shows the difference between feedforward neurons and recurrent neurons.

Assuming a RNN is processing sequential data $x_{1:T} = (x_1, \ldots, x_T)$. Mathematically, a simple recurrent neuron is written as:

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \tag{3.5}$$

$$y_t = \sigma_y(W_y h_t + b_y) \tag{3.6}$$

Here, x_t, h_t and y_t are the input data, hidden vector/ memory and output at time t. W, U and b are the weights and biases that are shared over all the time inputs. σ_h and σ_y are the activation functions.

Eventhough RNNs are designed for sequences, they are found to be inadequate for long sequences. This is because as the length of the sequence increases, backprogation suffer from vanishing or exploding gradients (Bengio et al., 1994). To over-

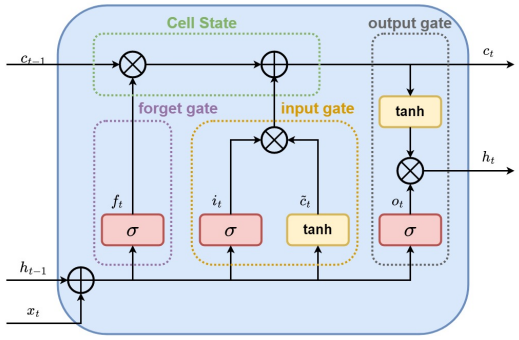


Figure 3.6: Block diagram of a LSTM cell

come this problem, two variants namely LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) have been developed, that are widely used. Moreover, due to its inherent sequential nature and backpropagation through time (BPTT), RNNs can't process each input in the sequence parallelly, leading to a slower training speed. To tackle this issue, modern variations of RNNs such as Linear Recurrent Unit (LRU) (Orvieto et al., 2023), minLSTM (Feng et al., 2024), minGRU (Feng et al., 2024), etc. have been developed that focuses on using parallel scan (Martin and Cundy, 2018) and linear sequential dependencies. We now describe the two major types of RNNs briefly, that we used in the thesis.

3.3.1 Long short-term memory (LSTM)

LSTM was first proposed by (Hochreiter and Schmidhuber, 1997), to handle the vanishing gradient problem of standard RNN. This was done by introducing gating mechanism that regulates the flow of information in and out of the memory. A simple LSTM cell is composed of an input gate, an output gate and a forget gate. The inputs to the cell are a) cell state - that holds the long term memory, b) hidden state - that holds the output of the previous point in time and c) input data at the current time step. Despite such an intricate architecture, it may still exhibit exploding gradient issues. A simple LSTM cell is shown in Figure 3.6.

The aim of input gate is to retain only the relevant input that needs processing by the LSTM cell. Similarly, the output gate is applied only to output the knowledge pertaining to the task at hand. In sequential data, sometimes past memory is required to processing the current data. Thus, a forget gate is used to refresh or retain the memory of the LSTM cell.

Mathematically, the functions of a LSTM are defined as:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{3.7}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{3.8}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{3.9}$$
$$\tilde{c}_t = \tanh_c(W_c x_t + U_c h_{t-1} + b_c) \tag{3.10}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.11}$$
$$h_t = o_t \odot \tanh_h(c_t) \tag{3.12}$$

Here, x_t, c_t and h_t are the input, cell state and hidden state of LSTM cell at time t respectively. The input to a LSTM cell consist of x_t, c_{t-1} and h_{t-1}. i_t, o_t and f_t are the activation vectors for input gate, output gate and forget gate respectively. W, U and b are the parameters of the cell. $\odot$ is the hadamard product.

3.3.2 Gated Recurrent Unit (GRU)

Motivated by the gated mechanism of LSTM, in (Cho et al., 2014), the authors proposed gated recurrent unit with a simpler architecture with less number of gates. Mathematically, a GRU is defined as:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \tag{3.13}$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \tag{3.14}$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{3.15}$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \hat{h}_t \tag{3.16}$$

Here, x_t and h_t are the input vector and output vector at time t. z_t and r_t are the activation vector for forget gate and input gate, respectively. W, U and b are the parameters of the cell.

3.4 Video Representation

Typically, video is considered as a 3D input of $H \times W \times T$ dimension, where $H \times W$ denotes the visual frame at any given time and T denotes the total number of frames in the video. This 3D input contains a huge amount of information with a lot of redundancy and thus, any good model needs to capture only relevant information ignoring the redundancy effectively. One widely known approach is to apply convolutions over the video. Depending upon the application and goal, this convolution could be a 2D or a 3D operation of $h \times w$ or $h \times w \times t$, respectively. In the following, we discuss a few representation learning approaches that we utilize in the thesis.

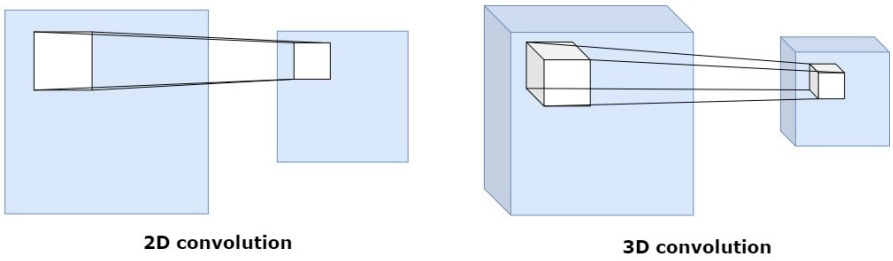


Figure 3.7: Comparison of 2D and 3D convolutions. The 2d convolution consists of a 2d kernel in contrast to 3d kernels for 3d convolution. The kernel may have different sizes in each dimension. Typically the time dimension of incase of a video.

3.4.1 Neural Representation: 2DCNN vs 3DCNN

Convolutional Neural Networks (CNNs) can be categorized based on the convolutional kernel's dimension. A 2D CNN uses 2D convolutional kernels in contrast to a 3D CNN which uses 3D convolutional kernels. These kernels are applied on a single frame or a clip of the video respectively. Because a 2D CNN is applied on a single frame as input, they inherently fail to leverage temporal context. Despite the capability of the 3D convolution kernels to handle 3D video input, 3DCNNs are constrained by GPU memory while processing very long videos and as such may need to be divided temporally in sub-clips as seen in (Richard et al., 2018; Vicol et al., 2018). Figure 3.7 shows the two different convolution operations.

There exist various efficient architectures such as Alexnet (Krizhevsky et al., 2012), VGGnet (Simonyan and Zisserman, 2014b), etc. for 2D representation and I3D (Carreira and Zisserman, 2017), Slowfast (Feichtenhofer et al., 2019), etc. for 3D representation.

Due to their limited receptive field of the convolution in the CNNs, they are limited in capturing global relations. This has paved way for global attention based transformer architectures such as ViViT (Arnab et al., 2021), Swin Transformer (Liu et al., 2021, 2022), etc. We will now briefly discuss the few CNNs and architectures used for video representation in the thesis.

3.4.2 Two-Stream Networks

The two-stream network architecture (Simonyan and Zisserman, 2014a) is inspired by the two-stream hypothesis for biological visual cortex (Goodale and Milner, 1992). The hypothesis indicates that the brain has separate pathways for static objects and motion-related information. The two-stream networks mimic the hypothesis by applying two different networks (or streams) to process static spatial information and temporal/motion information respectively as shown in Figure 3.8.

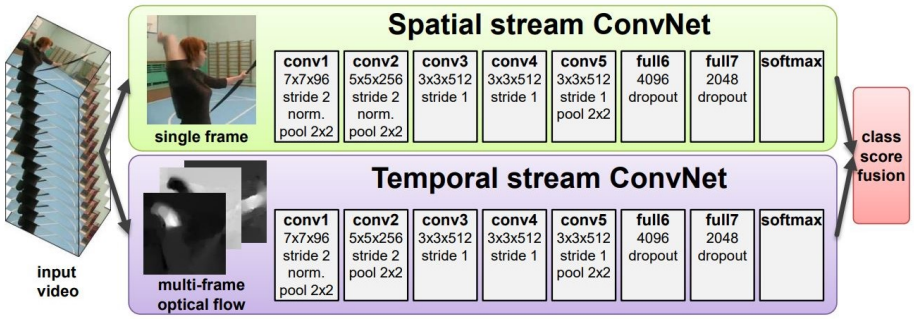


Figure 3.8: Overview of the two stream architecture (Simonyan and Zisserman, 2014a). The spatial stream ingests a single frame to parse the spatial information whereas the temporal stream focuses on an optical flow field to capture the motion information.

The input is a clip $H \times W \times T$ (a set of T consecutive frames of height H and width W), where the center frame is passed into the spatial stream to parse the spatial information. The whole set of T consecutive frames is then used to compute optical flow fields of size $H \times W \times T$. These optical flow fields are then passed as input for the temporal stream to process motion information. Once the output of each stream is computed, representations from both are fused together to make a spatiotemporal representation. These fused representations are used for predictions. Many sophisticated fusion strategies were later developed that helped to enrich the spatio-temporal representation.

3.4.3 I3D

Inflated 3D or I3D (Carreira and Zisserman, 2017) is a 3D CNN that extends a 2D architecture to perform action recognition. The core idea is to inflate the 2D square filters of size $N \times N$ by adding a new dimension to make them 3D cubic filters of $N \times N \times N$. This is done by repeating the weights of the 2D filters N times along the time dimension. This enables the architecture to bootstrap the new 3D filters from pretrained 2D filters. In other words, they initialize the 3D filters with the weights of the 2D filters trained on large image datasets like ImageNet (Deng et al., 2009) for significantly improved performance. The initialization also includes that the average and max-pooling layers be the same for the 2D case as the 3D case. Thus, if one repeats a single image N times to make a static video, the output of 3D architecture remains the same as that of the single image with 2D architecture. I3D also suggests the kernel should not be symmetric in time rather depend upon the frame rate and image dimensions. This is because if the receptive field are large in time, it may conflate the edges of different moving objects, messing up the early

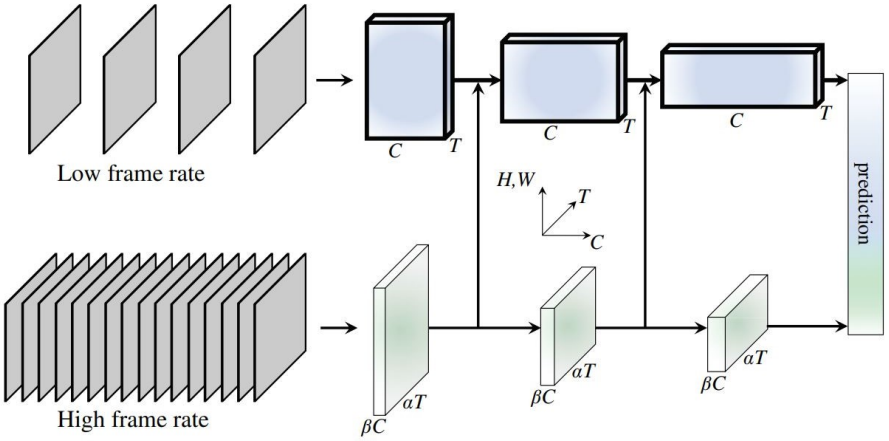


Figure 3.9: Overview of the Slowfast architecture. The slow pathway operates at a lower frame rate with to capture static content. The fast pathway, in contrast, operates at a higher frame rate focusing on identifying the dynamic content of a video. (Feichtenhofer et al., 2019).

edge features. Similarly, if the receptive field are small, it may not capture scene changes and dynamics as well. To further incorporate motion information, I3D can be deployed in two stream architecture where one stream ingests N set of consecutive frames and the other stream takes the corresponding set of N optical flow fields.

3.4.4 Slowfast

A standard two-stream architecture captures spatial and motion information via two similarly arranged parallel streams with different inputs of RGB frames and optical flow field. Though, effective they require explicit computation of the optical flow field. The use of a similar architecture for both RGB frames and optical flow field limits their ability to harness motion information effectively. These limitations are overcome by a 3DCNN architecture called Slowfast (Feichtenhofer et al., 2019) as shown in Figure 3.9.

At the heart, Slowfast uses two parallel pathways (similar to two streams), each designed to process different aspects of the video. The slow pathway operates at a lower frame rate with higher image resolution, enabling it to capture static or slow-moving content and provide a comprehensive understanding of the scene. In contrast, the fast pathway processes the video at a higher frame rate with reduced image resolution, allowing it to focus on capturing dynamic, fast-moving content. This is partially inspired by the retinal ganglion in primates (Derrington and Lennie, 1984; Van Essen and Gallant, 1994) that has P-cells and M-cells. P-cells operate on a low temporal frequency that helps in recognizing spatial details, whereas M-cells

operate at a high temporal frequency and are responsive to swift changes.

The Slow pathway uses a large temporal stride (i.e. number of frames skipped per second) τ resulting in a sparse sampling of frames in time. The Fast pathway uses a much smaller temporal stride $\frac{\tau}{\alpha}$ resulting in a dense temporal sampling of frames. To manage the computations, the Fast pathway is kept lightweight by using $\frac{1}{\beta}$ times smaller channel size compared to the slow pathway. As a result, the Fast pathway requires 4x less computing than the Slow pathway despite having a higher temporal frequency.

3.5 Localisation Task

The localization task in computer vision deals with predicting class and their location in visual content. For example, an object recognition algorithm (Ren et al., 2015; Redmon et al., 2016) deals with recognizing the object as well as localizing the object via a set of 4 continuous numbers namely, x and y coordinates & the height and width to draw a bounding box around the object. The location information is obtained by performing a regression task. Person detection is a specific type of object localization where the object to be localized is a human. Person localization can be performed on a single frame or a set of frames capturing the action in consideration. In the following section, we briefly introduce two localization algorithms related to the thesis.

3.5.1 Fast RCNN

The basis of Fast RCNN (Girshick, 2015) lies in RCNN (Girshick et al., 2014). In RCNN, the authors use selective search to generate 2000 region proposals that are warped and passed through a convolutional network for classification and bounding box regression. However, the speed of detection is bottlenecked by the convolutional computations for a large number of proposals.

To overcome the multiple CNN computations for each proposal, Fast RCNN uses a single convolutional network for the whole image and generates a feature map. The authors parallelly identify the proposals through selective search and generate a fixed-size vector using a RoI pooling layer from the feature map followed by a fully connected layer. The overview of the approach is shown in Figure 3.10

3.5.2 Faster RCNN

For faster RCNN (Ren et al., 2015), similar to Fast RCNN, an image is passed through a convolutional network to generate a feature map. Typically, the computation of proposals using the selective search algorithm of RCNN (Girshick et al., 2014) and Fast RCNN (Girshick, 2015) is expensive and results in low recall. Thus, the authors for Faster RCNN proposed using a proposal network over the feature maps.

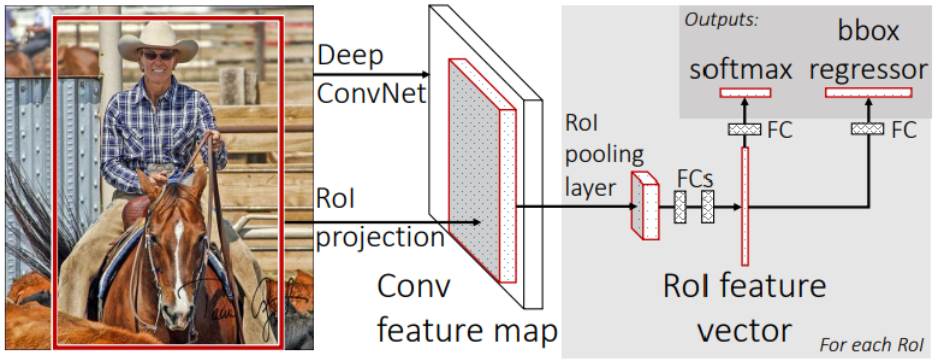


Figure 3.10: Overview of the Fast-RCNN architecture. An input image and multiple proposals or RoIs are input into a fully convolutional network. Each proposal is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers. (Girshick, 2015).

This proposal network generates quick and accurate proposals for object detection and localization. The novel region proposal network uses a sliding window over the feature map. This feature map generates multiple proposals based on fixed-shaped k-anchor boxes. Each proposal has 4 values for location and 2 values for objectness score. The RoI pooling for object detection and localization remains the same as Fast-RCNN. The overall architecture of the approach is shown in Figure 3.11a and the details of the region proposal are shown in Figure 3.11b.

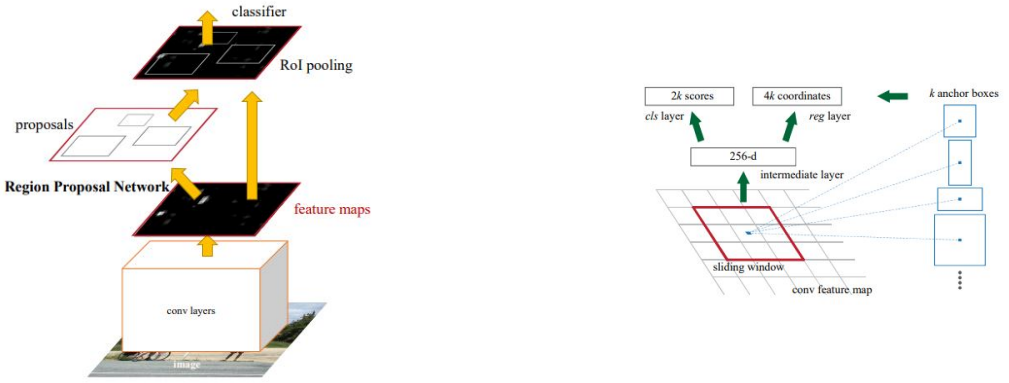


Figure 3.11: a) Overview of the Faster-RCNN architecture. An input image is passed through a fully convolutional network to generate feature maps. It uses region proposal and RoI pooling to perform object detection and localization (Ren et al., 2015). b) Overview of the Region Proposals. Region Proposals uses a sliding window over the feature map. For each sliding window location, k anchor boxes are generated characterized by objectness score and its coordinates.

3.6 Linear programming and optimization

Linear programming (LP) is an optimization method to achieve a solution that aims to maximize or minimize any linear mathematical function subject to any given linear constraints. The linear constraints could be defined as linear equality or inequality. A standard form of such problems is defined as:

$$\begin{aligned} \arg\min_{x} \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{3.17}$$

Here, x are the variables for which the optimal solution is to be found such that the objective function $c^T x$ is minimized. c is the coefficient vector. $Ax \leq b$ and $x \geq 0$ are the constraints over the solutions for x. It is also known as the *feasible region* or *convex polytope*. A is a given weight matrix for the constraints. Figure 3.12 shows a simple plot of the feasible region and the multiple constraints for two variables.

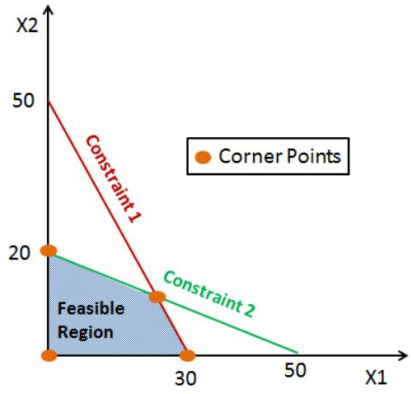


Figure 3.12: A simple plot of feasible region and constraints (edges) resulting in corner points/vertices for two variables

A standard linear objective function without constraints is a convex function. Thus, an optimal solution will definitely exist. However, due to the linear constraints, such an objective function might have no optimal solutions. This could be due to the fact that a) the constraints might result in a no-feasible solution and b) the direction of the objective gradient is unbounded due to constraints.

Linear programming is widely used in various fields such as business, economics, and many engineering problems. One of the widely used applications of linear programming lies in cost-effective planning and resource utilization in industries such as transportation, energy, telecommunications, and manufacturing to maximize output.

3.6.1 Special cases of Linear Programming

The optimal solution for the variable x is typically considered to be real numbers. i.e. $x \in \mathbb{R}$. However, two special cases of linear programming exist:

1. *Integer programming*: In these types of linear programming, the optimal solution needs to be an integer. In other words, $x \in \mathbb{I}$. Due to this constraint, these integer programs may not be solved efficiently and are considered NP-hard. Graph coloring problems are a type of Integer programming. ***0-1*** *programs or binary programming* are a special case of Integer programming where the optimal solution can only have values 0 or 1. Despite being NP-hard many algorithms exist that are able to generate approximate solutions in polynomial time (Hansen et al., 2013; Lee and Sidford, 2015).

2. *Mixed integer programs*: If the optimal solution for some of the variables (x) are integers and the rest could be real numbers, such integer programs are called *Mixed integer programs*. They are also NP-hard problems.

3.6.2 Algorithms

Linear programs such as the above can be solved by various algorithms. All the set of such algorithms can be divided into two categories as follows:

1. *Simplex method*: The simplex method (Dantzig, 1990) was developed by George Dantzig. These categories of methods typically follow an iterative process that relies on mathematical calculations and logical reasoning over the corner points to find the optimal solution to a linear programming problem.

   Despite its simplicity, the simplex method has the worst time complexity. In some cases, it may result in cycles and result in non-polynomial time computations.

   Some recent approximate methods such as (Hansen et al., 2013) are based on simplex methods.

2. *Interior point method*: In contrast to the simplex algorithm that achieves an optimal solution by iterating over the edges between the corner points of a feasible region, the interior-point methods move through the interior of the feasible region.

   The most famous interior point method is called *Karmarkar's algorithm* (Karmarkar, 1984; Adler et al., 1989) which was introduced by Narendra Karmarkar. It is able to find the optimal solution in a polynomial time. If n denotes the number of variables and L denotes the number of bits of input, then the time complexity of Karmarkar's algorithm is found to be $\mathcal{O}(n^{3.5}L)$.

The recent and updated versions of Karmarkar's algorithm such as (Lee and Sidford, 2015) are able to achieve better worst-case time complexity of $\mathcal{O}(n^{2.5}L)$.

Despite polynomial time computations, it is sometimes beneficial to achieve approximate solutions in a faster time. This is critical in the case of Binary Linear programming that is NP-hard. Many such approximate solutions are based on *branch and bound algorithms* (Mitra, 1973) or *cutting plane methods* (Gilmore and Gomory, 1961, 1963) or combinations of their-of. Multiple commercial solvers such as CPlex[1] or Gurobi[2] exist that generates exact or approximate solutions for practical usage.

[1] https://www.ibm.com/docs/en/icos/20.1.0?topic=cplex-users-manual
[2] https://www.gurobi.com/

Chapter 4

Structural Recurrent Neural Network (SRNN) for Group Activity Analysis

In this chapter, we present a solution for group activity recognition that uses structural recurrent neural networks (SRNN) to jointly identify individual actions and overall group activity. Understanding group behaviors requires capturing the spatial and temporal interactions among individuals within the group. Our solution consists of two SRNN variants designed to model these relationships explicitly using multiple RNNs and trained together using a single loss function.

Individual Contribution

The following chapter is based on the publication (Biswas and Gall, 2018):

Structural Recurrent Neural Network (SRNN) for Group Activity Analysis
Sovan Biswas, and Jürgen Gall.
IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2018.

This publication was done by Sovan Biswas and Jürgen Gall provided scientific guidance and supported this work with very valuable feedback and suggestions.

Contents

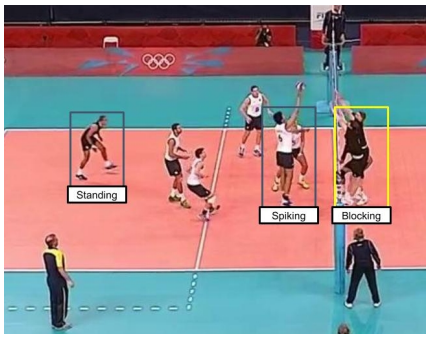


Figure 4.1: A frame labeled as group activity “Left Spike" and bounding boxes around each team player are annotated in the dataset with individual actions (Ibrahim et al., 2016b).

4.1 Introduction

Activity analysis has been of great interest in computer vision since decades. In recent years, deep learning approaches such as (Carreira and Zisserman, 2017; Ji et al., 2013; Simonyan and Zisserman, 2014a; Singh et al., 2016; Karpathy et al., 2014) have been proposed to recognize activities in videos. Most of these approaches, however, focus on single person activity analysis and estimate only one activity per video clip. Similar to other recent works (Ramanathan et al., 2016; Deng et al., 2016; Alahi et al., 2016; Shu et al., 2017), the goal of this chapter is to understand and analyze the actions of individuals and their interactions, and subsequently use them for predicting the group activity.

Recent deep learning approaches for group activity analysis such as (Ibrahim et al., 2016a; Shu et al., 2017) use a multi-level hierarchy of recurrent neural networks (RNNs) for group activity recognition. In these approaches, the lower level RNNs focus on understanding and modeling the actions of individuals and the higher level RNNs in the architecture model the group activity. These approaches are trained using a two-step process. The first step focuses on improving the recognition of the actions of each individual independently and the subsequent step focuses on recognizing the group activity given the recognized actions of the individuals.

Apart from the hindrance of two-step training, these methods lack the capability of capturing interactions between individual persons when present within a group. For example, in a volleyball game as shown in Figure 4.1, a person from the team on the left-hand side of the volleyball court performs the individual action “spiking" whereas the player from the opponent team performs a “blocking" action. Looking

only at the individuals makes it difficult to distinguish between the individual actions, but there is a strong correlation between the two activities. Similarly, when walking in a crowd, people move and walk in various directions just to avoid colliding with each other. In general, the action of an individual in a group is influenced by the actions of the other individuals in the group. This phenomenon not only provides context that helps to recognize the individual actions but also provides a key information about the group level actions such as in the case of volleyball as shown in Figure 4.1. Thus, there is an imperative need to analyze interactions between individuals and to capture the influence over time when analyzing a group of humans.

The main focus of the chapter is to harness such interactions within a group to improve the recognition of the group activity as well as the individual actions. To this end, we build on the recently proposed structural recurrent neural network (SRNN) (Jain et al., 2016) which has the unique capability of capturing interactions as contextual information using an interconnected set of RNNs. While in (Jain et al., 2016), the number of nodes and edges and therefore the number of RNNs is constant, we extend the approach to handle a varying number of nodes and edges as it is required for analyzing group activities.

The rest of the chapter is structured as follows. We start with a brief introduction of SRNNs in Section 4.2. In Section 4.3, we introduce two variants of an SRNN. We then evaluate the two variants in Section 4.4 and conclude with a summary in Section 4.5.

4.2 Structural RNN

Recurrent neural networks are very effective in modeling temporal sequences. In case of a single person, features f^t are extracted at each frame and used as input for an RNN to predict the action classes y^t over time. However when in a group, a person performs an action based on its interaction with other persons and the group objective. So, a single recurrent neural network is incapable of capturing the interactions and group dynamics, thus reducing its effectiveness. For solving similar problems, in (Jain et al., 2016), the authors proposed an interconnected set of recurrent neural networks that not only captures the individual behavior over time but also integrates the interactions between the individuals through edges.

An example of an SRNN is illustrated in Figure 4.2. It consists of three nodes v_i, v_j, and v_k and the goal is to predict for each node the class labels over time, which are denoted by $y^t_{v_i}$, $y^t_{v_j}$, and $y^t_{v_k}$, respectively. Each node is modeled by an RNN, termed nodeRNN. It takes as input some features $f^t_{v_i}$, which are extracted for a node v_i, but also the output of RNNs that model the interactions with other nodes. The second type of RNN is termed edgeRNN. The edgeRNNs take as input some features $f^t_{e_{ij}}$ based on the spatio-temporal relation between two nodes v_i and v_j and predicts

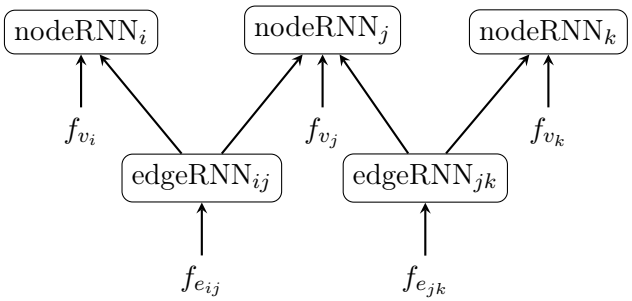


Figure 4.2: Feedforward network of a Structural RNN (SRNN) when trained with respect to node labels.

a latent representation $h^t_{e_{ij}}$, which is forwarded to the corresponding nodeRNNs. The advantage of such an SRNN is that the nodeRNNs and the edgeRNNs can be trained jointly such that the prediction of the node labels depends not only on the features that are extracted for each node but also on the interactions between the nodes.

4.3 Group Activity Analysis

In this section, we will first briefly introduce the problem in Section 4.3.1. This is followed by introducing two different variants of an SRNN for group activity recognition in Section 4.3.2.

4.3.1 Problem Formulation

Our objective is to predict jointly the group activity label y^t_g of a group as well as the action label $y^t_{v_i}$ for each individual v_i of the group over time. We assume that the bounding box for each person has been already extracted and we compute features for each individual person $f^t_{v_i}$ and for each edge $f^t_{e_{ij}}$ between two individuals. The features are described in Section 5.2.3 and we denote the set of all node and edge features for a frame by F^t. In order to learn the parameters θ of the models which are described in Section 4.3.2, we minimize the loss

$$\arg\min_{\theta} \Big[L\left(\psi_g\left(F^t;\theta\right), y^t_g\right) + \frac{1}{n}\sum_{i=1}^{n} L\left(\psi_v\left(F^t;\theta\right), y^t_{v_i}\right)\Big], \tag{4.1}$$

where L denotes the cross-entropy loss, $\psi_g(.)$ the prediction function of the model for the group activity, and $\psi_v(.)$ the prediction function for the actions of the individuals.

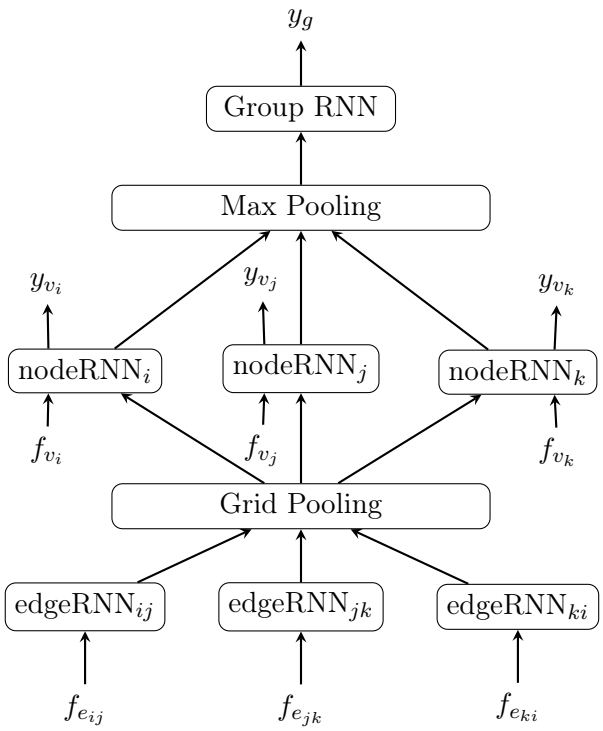


Figure 4.3: SRNN-MaxNode: Feedforward SRNN where max pooling is performed over the nodeRNNs. The nodeRNNs are enriched using the output of the edgeRNNs using a novel grid pooling approach.

4.3.2 SRNN for Group Activity Analysis

The proposed group activity recognition approach is formulated as a two-level hierarchy of recurrent neural networks similar to (Ibrahim et al., 2016a; Shu et al., 2017). The lower level predicts individual actions followed by the higher level recurrent network that estimates the group activity. In Section 4.3.2.1 and Section 4.3.2.2, we discuss two SRNN variants that jointly estimate the group activity and the individual actions by modeling the interactions between individuals. The two SRNNs are shown in Figure 4.3 and Figure 4.4.

4.3.2.1 SRNN-MaxNode

As shown in Figure 4.1, when a person on the left hand side jumps to perform the action “spiking", the opponents jump to block the spike, resulting in the action “blocking" across the volleyball net. This is an example where persons in a group perform contextual actions. Structural RNNs are an efficient approach to model such relations.

As discussed in Section 4.2, SRNNs are a hierarchy of RNNs consisting of

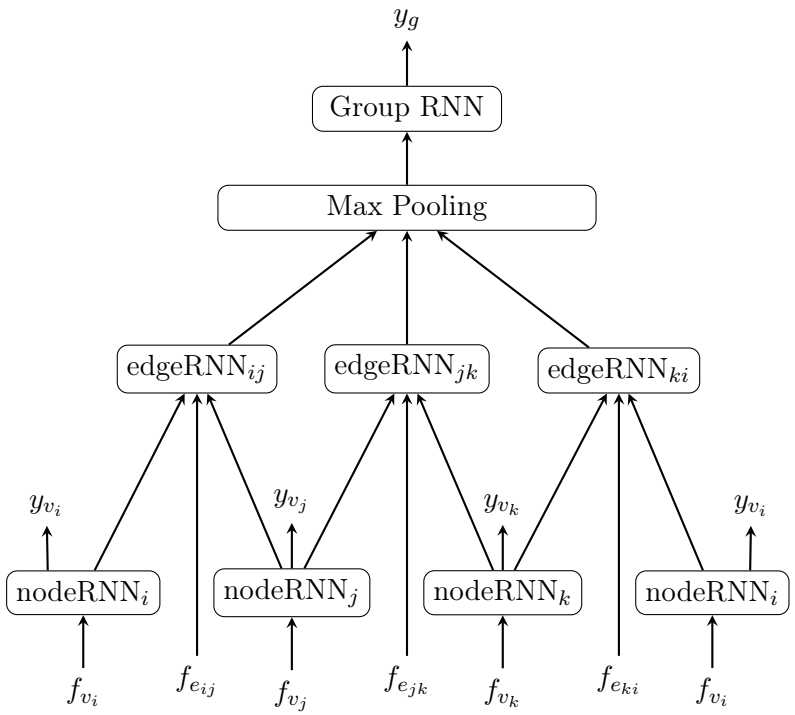


Figure 4.4: SRNN-MaxEdge: Feedforward SRNN where max pooling is performed over the edgeRNNs. The edgeRNNs are enriched using the output of the nodeRNNs.

edgeRNNs and nodeRNNs. The first variant that we propose for group activity analysis is shown in Figure 4.3. The lowest level consists of edgeRNNs that model interactions between two individuals based on their relative position, which is encoded by the feature vector $f^t_{e_{ij}}$. The output of the edgeRNNs is feedforwarded to the nodeRNNs. The number of individuals, however, varies in a group and each individual might have a different number of neighbors and in very dense crowds the number of neighbors could be very high. The approach proposed in (Jain et al., 2016) cannot handle such cases since it concatenates the features, assuming that the number of edges and nodes is constant. To address this problem, we propose a grid pooling layer that combines for a node v_i the output from all edgeRNNs e_{ij} based on the position of the neighboring persons v_j in a prescribed grid. The prescribed grid regions are arranged as shown in Figure 4.5. If several neighbors are in the same grid cell, we sum the output of the edgeRNNs instead of averaging them. This means that the values are usually larger when more persons are in a cell. The grid pooling provides for each node v_i features for 8 cells that are then concatenated with additional CNN features $f^t_{v_i}$, which are extracted from the frame t for each person. The output of the nodeRNNs is used in two ways. First, the action class $y^t_{v_i}$ of the person v_i is predicted using an additional softmax layer. Second, the group activity

is estimated similar to (Ibrahim et al., 2016a) by max-pooling the outputs of the nodeRNNs of a group at a time instant t, which is then used as input for an RNN that predicts the group activity y_g^t over time.

In summary, the SRNN-MaxNode model is defined as follows:

$$\begin{aligned} h_{e_{ij}}^t &= \text{RNN}_e(h_{e_{ij}}^{t-1}, f_{e_{ij}}^t) \\ h_{C_i}^t &= \sum_{j \in S_{C_i}} h_{e_{ij}}^t \\ h_{e_i}^t &= [h_{L^i}^t \dots h_{Q_4^i}^t] \\ h_{v_i}^t &= \text{RNN}_v(h_{v_i}^{t-1}, h_{e_i}^t, f_{v_i}^t). \end{aligned} \tag{4.2}$$

While $h_{e_{ij}}^t$ denotes the output from the edgeRNN for the nodes v_i and v_j at frame t, S_{C_i} denotes the set of neighboring nodes of v_i in the cell $C_i \in \{L^i, R^i, A^i, B^i, Q_1^i, Q_2^i, Q_3^i, Q_4^i\}$, which are the grid regions for v_i as shown in Figure 4.5. The accumulated values for each cell $h_{C_i}^t$ are then concatenated to the vector $h_{e_i}^t$ and the output of the nodeRNN is denoted by $h_{v_i}^t$.

The full architecture can thus be defined as:

$$\begin{aligned} h_{v_i}^t &= \text{SRNN}(f_{v_i}^t, f_{e_{ij}}^t) \\ y_{v_i}^t &= \phi_v(h_{v_i}^t, W_v) \\ h_p^t &= \max(h_{v_1}^t \dots h_{v_n}^t) \\ h_g^t &= \text{RNN}_g(h_g^{t-1}, h_p^t) \\ y_g^t &= \phi(h_g^t, W_g), \end{aligned} \tag{4.3}$$

where $h_{v_i}^t$ denotes the output after the SRNN (4.2) and W_v denotes the weights used in the softmax function $\phi_v(.)$ to predict the individual actions $y_{v_i}^t$. h_p^t denotes the max-pooled representation over the complete group and h_g^t denotes the output of the group RNN, which is then used by a softmax function $\phi(.)$ with weights W_g to predict the group activity y_g^t.

4.3.2.2 SRNN-MaxEdge

While the SRNN in Figure 4.3 uses the edgeRNNs to provide contextual information for the nodeRNNs, we also compare it to an SRNN where the nodeRNNs are at the lowest level of the hierarchy. In this case, the max pooling is not performed over the nodeRNNs but over the edgeRNNs and an additional grid pooling is not required since each edge consists of two nodes. We denote the second variant, which is shown in Figure 4.4, SRNN-MaxEdge.

For SRNN-MaxEdge, the lower level of the hierarchy consists of nodeRNNs that predict the individual actions based on the individual CNN features $f_{v_i}^t$. The output from the nodeRNNs is forwarded to the corresponding edgeRNNs, which also take

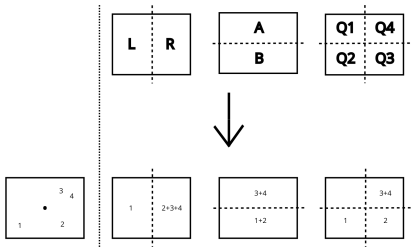


Figure 4.5: Grid pooling. Left: 1, 2, 3 and 4 denote four persons in the neighborhood of the person •. Right: We define three grid structures (top) and we sum the outputs of the edgeRNNs where the neighbors of • are in the same cell. We then concatenate the features of the eight cells. If a cell is empty, the feature vector is set to zero.

the edge features $f^t_{e_{ij}}$ as input. The output of the edgeRNNs at a time instant t is max pooled and then used as input for an RNN that predicts the group activity y^t_g over time.

In summary, the SRNN-MaxEdge model is defined as follows:

$$\begin{aligned} h^t_{v_i} &= \text{RNN}_v(h^{t-1}_{v_i}, f^t_{v_i}) \\ y^t_{v_i} &= \phi_v(h^t_{v_i}, W_v) \\ h^t_{e_{ij}} &= \text{RNN}_e(h^{t-1}_{e_{ij}}, h^t_{v_i}, h^t_{v_j}, f^t_{e_{ij}}), \end{aligned} \tag{4.4}$$

where $h^t_{v_i}$ denotes the output of the nodeRNN for person v_i at a given time t and W_v are the weights used in the softmax function $\phi_v(.)$ to predict the individual action $y^t_{v_i}$. $h^t_{e_{ij}}$ denotes the output of the edgeRNNs.

The full architecture can thus be defined as:

$$\begin{aligned} h^t_{e_{ij}} &= \text{SRNN}(f^t_{v_i}, f^t_{e_{ij}}) \\ h^t_p &= \max\left(h^t_{e_{12}} \dots h^t_{e_{mn}}\right) \\ h^t_g &= \text{RNN}_g(h^{t-1}_g, h^t_p) \\ y^t_g &= \phi(h^t_g, W_g), \end{aligned} \tag{4.5}$$

where $h^t_{e_{ij}}$ denotes the output from the SRNN (4.4). While h^t_p denotes the max pooled representation over all edges in the group, h^t_g denotes the output of the group RNN, which is then used by a softmax function $\phi(.)$ with weights W_g to predict the group activity y^t_g.

Method	Group Activity Recognition	Individual Action Recognition
Hierarchical LSTM (1 group)	70.3%	-
Hierarchical LSTM V1 (1 group)	68.37%	**76.32%**
Hierarchical LSTM V2 (1 group)	73.89%	**76.32%**
Hierarchical LSTM V3 (1 group)	**74.01%**	75.96%
Hierarchical LSTM (2 groups)	81.9%	-
Hierarchical LSTM V1 (2 groups)	78.37%	**76.32%**
Hierarchical LSTM V2 (2 groups)	81.33%	**76.32%**
Hierarchical LSTM V3 (2 groups)	**83.12%**	75.96%

Table 4.1: Comparison of various variations of the Hierarchical LSTM (Ibrahim et al., 2016b) using Alexnet features.

4.4 Experiments

4.4.1 Datasets

We evaluate our framework on the recently introduced volleyball dataset (Ibrahim et al., 2016b). This dataset has 55 volleyball game video sequences with 4830 labeled frames, where each player is labeled and subsequently annotated with the bounding box. Each player performs one of the 9 individual actions resulting in one of the 8 group activity labels. Furthermore, the whole dataset is divided into non-overlapping sets of 24 sequences for training, 15 sequences for validation and the remaining sequences are used for testing. Similar to (Ibrahim et al., 2016a; Shu et al., 2017), we have used both training and validation sequences for training. Since not all frames are annotated by bounding boxes, the Dlib tracker (King, 2009) is used to propagate the ground-truth bounding boxes to the unannotated frames.

4.4.2 Implementation Details

For the RNNs, we use standard LSTMs (Hochreiter and Schmidhuber, 1997) and the implementation is done using the Tensorflow library. At the node level, each LSTM is connected with a deep convolutional network such as Alexnet (Krizhevsky et al., 2012) or VGG 16 (Simonyan and Zisserman, 2014b) to compute the visual features f_{v_i} based on the annotated and tracked bounding boxes of the persons. Similar to (Ibrahim et al., 2016a; Shu et al., 2017), we initialize the CNNs by a model that has been pre-trained on ImageNet. During training, we fine-tune only the last two fully connected layers of the CNNs.

The edge features $f^t_{e_{ij}}$ model spatio-temporal relations between the bounding boxes of two persons. We take the center of each bound box and compute the difference vector (dx, dy). We then compute the basic distance values ($|dx|$, $|dy|$, $|dx+dy|$,

$\sqrt{(dx)^2 + (dy)^2}$) and add the direction of the translational vector ($arctan(dy, dx)$, $arctan2(dy, dx)$). To further enhance these simple 6 interaction features, we also compute the difference of the 6 features between two consecutive time frames, which results in 6 additional features. We finally compute the 12 features not only for frame t, but also for the neighboring frames $t-1$ and $t+1$ to capture some short temporal information. All features are concatenated to obtain a 36 dimensional feature vector.

The training is performed in two stages. In the first stage, the nodeRNNs are trained independently using individual actions and the cross-entropy as loss function. For this stage, we have used a batch size of 36 for our experiments. In the next step, we train the whole architecture by minimizing the loss (4.1). We use Adam as optimizer with a learning rate of 0.00001. During training, the parameters of the nodeRNNs and the last two layers of the CNN are updated as well. The nodeRNNs have 3000 hidden units and the group RNN has 2000 hidden units. The number of nodes for the edgeRNNs differs between the SRNN-MaxNode and the SRNN-MaxEdge due to differences of the input features. While the edgeRNNs in the SRNN-MaxNode take as input the low dimensional features $f^t_{e_{ij}}$, the edgeRNNs in the SRNN-MaxEdge use the additional output of two nodeRNNs as input. For the edgeRNNs in the SRNN-MaxNode, we use therefore only 30 hidden units whereas 1000 hidden units are used for the edgeRNNs in the SRNN-MaxEdge. Since the two variants differ in memory consumption and the GPU memory is limited, we use a batch size of 30 for SRNN-MaxNode and a batch size of 16 is used for SRNN-MaxEdge.

In accordance with other approaches Hierarchical LSTM (Ibrahim et al., 2016a), CERN (Shu et al., 2017) and Social Scene (Bagautdinov et al., 2017), we have also performed experiments where we divide the individuals in two groups. For this, we use the same approach as in (Ibrahim et al., 2016b).

4.4.3 Experimental Evaluation

4.4.3.1 Variation of Hierarchical LSTM

As the proposed approaches are inspired by Hierarchical LSTMs (Ibrahim et al., 2016b), we have first performed a few baseline experiments to evaluate versions of the Hierarchical LSTM (Ibrahim et al., 2016b):

- 2-layer LSTMs (V1): This is similar to (Ibrahim et al., 2016b) with a two-step training for person level RNNs followed by training of the group level RNN given the pre-trained person RNNs. Unlike (Ibrahim et al., 2016a), the group level takes as input only the output of person RNNs and does not use any additional CNN features.

- 2-layer LSTMs (V2): Reimplementation of (Ibrahim et al., 2016b).

Method	Group Activity	Individual Action Recognition
Hierarchical LSTM (1 group)	70.3%	-
Hierarchical LSTM V3 (1 group)	74.01%	75.96%
CERN (1 group)	73.5%	69%
SRNN-MaxNode (1 group)	**74.39%**	**76.65%**
SRNN-MaxEdge (1 group)	68.39%	76.03%
Hierarchical LSTM (2 groups)	81.9%	-
Hierarchical LSTM V3 (2 groups)	83.12%	75.96%
CERN (2 groups)	83.3%	69%
SRNN-MaxNode (2 groups)	**83.47%**	**76.65%**
SRNN-MaxEdge (2 groups)	79.86%	76.03%
Social Scene (2 groups)	**89.9%**	**82.4%**

Table 4.2: Comparison of the proposed SRNN approaches to the state-of-the-art.

Feature	Method	Group Activity	Individual Action Recognition
Alexnet	H. LSTM V3 - (1 group)	74.01%	75.96%
	SRNN-MaxNode - (1 group)	**74.39%**	**76.65%**
	SRNN-MaxEdge - (1 group)	68.39%	76.03%
VGG 16	H. LSTM V3 - (1 group)	70.34%	75.30%
	SRNN-MaxNode - (1 group)	71.20%	74.85%
	SRNN-MaxEdge - (1 group)	68.29%	75.96%
Alexnet	H. LSTM V3 - (2 groups)	83.12%	75.96%
	SRNN-MaxNode - (2 groups)	**83.47%**	**76.65%**
	SRNN-MaxEdge - (2 groups)	79.86%	76.03%
VGG 16	H. LSTM V3 - (2 groups)	81.34%	75.30%
	SRNN-MaxNode - (2 groups)	82.86%	74.85%
	SRNN-MaxEdge - (2 groups)	79.92%	75.96%

Table 4.3: Comparison of the proposed SRNN approaches with Hierarchical LSTM V3 using Alexnet or VGG 16 as CNN.

- 2-layer LSTMs (V3): This is similar to V1 but person RNNs and group RNN are jointly trained using the loss (4.1). As described in Section 5.2.3, the last two layers of the Alexnet are fine-tuned.

The accuracy of recognizing the group activity as well as the actions of the individuals is reported in Table 4.1. The results show that training the person RNNs and the group RNN jointly (V3) using the loss (4.1) improves the accuracy of the group activity also for the Hierarchical LSTM (Ibrahim et al., 2016b), but it slightly decreases the individual action recognition results. Dividing the individuals into two groups as in (Ibrahim et al., 2016b) improves the accuracy by a large margin due to the volleyball scenario where two teams play against each other.

4.4.3.2 Comparison to state-of-the-art

Table 5.5 compares the proposed SRNN approaches with the Hierarchical LSTM V3 that is trained with the same loss function and uses the same Alexnet CNN. While SRNN-MaxNode outperforms the Hierarchical LSTM both for group activity recognition as well as the recognition of the individual actions, SRNN-MaxEdge achieves a lower group activity accuracy than SRNN-MaxNode and Hierarchical LSTM. It shows that the max pooling over the nodeRNNs is better than pooling over the edgeRNNs on this dataset since the max pooling over the nodeRNNs forwards the features of the most important individual to the group RNN. This works for group activities as shown in Figure 4.1 very well since the group activity can be well inferred from the "spiking" person. Our proposed approach SRNN-MaxNode also outperforms the approach CERN (Shu et al., 2017), which also uses Alexnet features. However, the recent approach (Bagautdinov et al., 2017), which builds on the Inception-V3 CNN (Szegedy et al., 2016), achieves the highest accuracy on this dataset.

4.4.3.3 Impact of CNN architecture

We have also analyzed the impact of the CNN architecture and compare the used Alexnet CNN with the larger VGG 16 network. The results are reported in Table 4.3. The accuracy of the VGG 16 network decreases the accuracy of the Hierarchical LSTM as well as the proposed SRNN-MaxNode. For SRNN-MaxEdge the accuracy remains nearly the same. The decrease in accuracy might be due to overfitting, but it needs further investigation to analyze the impact of the used CNN model in more detail.

4.4.3.4 Impact of deep edge features

For the model SRNN-MaxNode, we use a low dimensional feature vector $f_{e_{ij}}^t$ that encodes simple spatio-temporal relations between two bounding boxes. We also

Edge feature	Group Activity Accuracy	Individual Action Recognition Accuracy
$f^t_{e_{ij}}$ (1 group)	74.39%	**76.65%**
$(f^t_{e_{ij}}, f^t_{v_i}, f^t_{v_j})$ (1 group)	**74.48%**	75.89%
$f^t_{e_{ij}}$ (2 groups)	**83.47%**	**76.65%**
$(f^t_{e_{ij}}, f^t_{v_i}, f^t_{v_j})$ (2 groups)	83.27%	75.89%

Table 4.4: Comparison of edge features using SRNN-MaxNode.

investigated if the accuracy can be improved when the edgeRNNs not only take $f^t_{e_{ij}}$ as input feature but also $f^t_{v_i}$ and $f^t_{v_j}$. Since this increases the dimensionality of the input feature from 36 to 8228 (4096+4096+36), we also increase the number of hidden units of the EdgeRNNs from 30 to 1000 to address the higher dimensionality. As shown in Table 4.4, adding $f^t_{v_i}$ and $f^t_{v_j}$ does not improve the accuracy. This is expected since the features $f^t_{v_i}$ are already added to the nodeRNNs and adding them twice does not provide additional information for the model.

4.5 Conclusions

In this chapter, we have proposed two variants of structural recurrent neural networks (SRNN) to recognize the actions of individuals as well as the activity of the entire group jointly. The advantage of the SRNN approach is that it explicitly models relations between individuals and all RNNs can be trained together using a single loss function. We evaluated the models on the Volleyball Dataset and showed that the SRNN model outperforms hierarchical LSTMs.

Chapter 5

Hierarchical Graph-RNNS for Action Detection of Multiple Activities

In this chapter, we introduce a method for multi-label action detection that uses hierarchical graph RNN that combines both temporal scene context and human interactions. The idea of using human interactions for spatio-temporal action detection was developed from concepts of group activity recognition discussed in the previous chapter.

Individual Contribution

The following chapter is based on the publication (Biswas et al., 2019):

Hierarchical Graph-Rnns for Action Detection of Multiple Activities
Sovan Biswas, Yaser Souri and Jürgen Gall.
IEEE International Conference on Image Processing (ICIP), 2019.

This publication was done by collaboration between Sovan Biswas and Yaser Souri. The core idea took shape as the result of multiple discussions between Sovan Biswas and Yaser Souri. The implementation and the experimentation was done by Sovan Biswas. Jürgen Gall provided scientific guidance and supported this work with very valuable feedback and suggestions.

Contents

5.1 Introduction

With the advent of deep neural networks and the availability of large datasets in the last decade, the performance of recent algorithms for action recognition has improved drastically. Annotating large amount of data, however, is very expensive. In particular for spatio-temporal action recognition and localization where multiple actions occur at the same time, action labels and bounding boxes would be required for each frame in order to learn a model using full supervision. The large-scale AVA 2.1 dataset (Gu et al., 2018) for recognizing and localizing multiple actions in videos, therefore, provides only temporally sparse annotations, i.e., the persons and the actions the people perform are annotated for only one frame per second. This requires to develop methods that can be trained with such sparse annotations (Gu et al., 2018; Sun et al., 2018; Girdhar et al., 2018; Stroud et al., 2018).

While these works take temporal information into account, they do not model the interactions of the individual persons. While some actions of different persons are uncorrelated, other actions refer to interactions between persons like 'talk to' and 'listen to'. There are also actions that are often performed by several persons at the same time like 'sit', 'stand', 'walk', or 'play instrument', but there are also actions that exclude each other. For instance, if a person 'drives' a car, it is very unlikely that the other persons in the car 'stand' or 'play an instrument'.

In this work, we, therefore, propose an approach that learns the relations of actions that occur at the same time and that can be learned using only sparse annotations. In order to address the sparseness of the annotations, we do not rely on tracked bounding boxes, which can be unreliable. Instead, we propose a hierarchical model that models the temporal scene context in the lower level and the relations of actions of the detected persons on the top level as illustrated in Figure 5.1. The temporal scene context is important since the relations of the actions depend on the scene. For instance, it matters if persons are inside of a moving car or a parking car. We model the scene context by a recurrent neural network (RNN) that uses I3D features (Carreira and Zisserman, 2017) as input. The RNN models the temporal context of the entire frames. At the top level of the hierarchy, we combine the hidden states of the scene context RNN with I3D features extracted for all detected persons in a frame. To learn the relations of the actions of all detected persons, we use a graph recurrent neural network (Scarselli et al., 2009; Gori et al., 2005; Li et al., 2016). The proposed model therefore learns the scene RNN and the graph RNN together.

We evaluate our approach on the large-scale AVA 2.1 dataset (Gu et al., 2018) where the approach achieves state-of-the-art results for action detection of multiple activities.

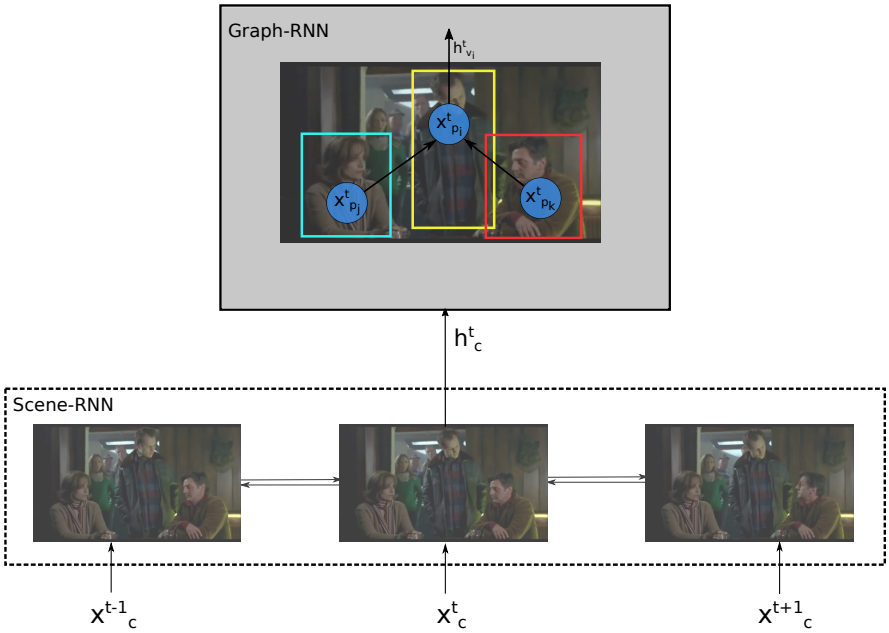


Figure 5.1: The proposed network jointly captures temporal context and the relations between different persons by a hierarchy. Scene RNNs, at the lower level, are used to model the temporal context of a scene. The Graph-RNN on top of it, models the relations of the actions of the detected persons.

5.2 Detection of Multiple Activities

As it is defined in the AVA 2.1 dataset (Gu et al., 2018), the goal of the approach is to estimate for a frame the bounding boxes of all persons performing an action and for each bounding box the action labels. In contrast to other datasets, each bounding box is annotated by multiple labels. For instance, a person might 'stand', 'carry/hold' an object, and 'listen to' another person at the same time.

5.2.1 Features

As illustrated in Figure 5.2, the proposed approach consists of two steps. We first detect the persons in a frame and then we use the hierarchical graph RNN to infer the action labels for each detected bounding box. For detecting the bounding boxes, we use Faster RCNN (Ren et al., 2015) with a ResNet architecture (He et al., 2016). The detector is fined-tuned on the dataset and the detections are performed in an action class agnostic way.

For the temporal context of a frame t, we extract I3D features (Carreira and Zisserman, 2017) for t and its neighboring frames. We only consider the frames t that are annotated in AVA, i.e., one frame per second, but a single I3D feature

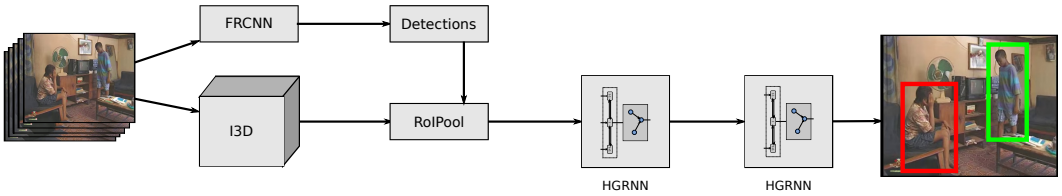


Figure 5.2: After detecting bounding boxes using Faster RCNN and pooling I3D features for each detected bounding box, the proposed hierarchical Graph-RNN predicts multiple class labels for each bounding box. In this example, the network detects two persons and infers the activities 'sit', 'watch (a person)', and 'listen to (a person)' for the red bounding box and 'stand', 'talk to (a person)', and 'watch (a person)' for the green bounding box.

is computed over a temporal context of 33 frames. This means that the temporal receptive fields of the extracted features for the annotated frames t and $t+1$ slightly overlap. Given l annotated frames before and after frame t, we get a sequence of $L = 2l + 1$ I3D features

$$x_c^{t-l}, \ldots, x_c^{t-1}, x_c^t, x_c^{t+1}, \ldots, x_c^{t+l}, \tag{5.1}$$

which corresponds to a temporal context of approximately L seconds for AVA.

We also use the detected bounding boxes for each sparsely sampled frame and use region pooling to extract I3D features per detected bounding box as in (Gu et al., 2018). This gives for each detected bounding box i in frame t another I3D feature vector that is denoted by $x_{p_i}^t$. We will use these features as input for the hierarchical graph RNN.

5.2.2 Hierarchical Graph RNN (HGRNN)

The proposed hierarchical graph RNN as illustrated in Figure 5.1 comprises two types of RNNs that are trained together. At the lower level, the scene RNN, which is described in Section 5.2.2.1, models the temporal scene context. The scene context estimated by the scene RNN is then used as input for the graph RNN, which models the relations of the actions performed by the detected persons. The graph RNN, which is described in Section 5.2.2.2, predicts then for each bounding box multiple action labels.

5.2.2.1 Scene RNN

The scene RNN takes all L scene features x_c^t as well as the features extracted for each detected person $x_{p_i}^t$ in all L frames as input and predicts a hidden state h_c^t only for frame t. Since it uses the frames before and after the frame t, we use a bidirectional RNN with GRUs. At this stage, we do not model any relations between the persons

and simply perform maxpooling over all features $x_{p_i}^t$ at each frame.

$$\begin{aligned} x_P^t &= maxpool_i(x_{p_i}^t) \quad (5.2) \\ x^t &= x_c^t \oplus x_P^t \\ h_c^t &= biGRU(x^t; h_c^{t-1}, h_c^{t+1}). \quad (5.3) \end{aligned}$$

While x_P^t denotes the maxpooled person feature for all detected persons in frame t, $\oplus$ denotes the concatenation of two vectors.

5.2.2.2 Graph RNN

Given i detected persons in frame t and the corresponding features $x_{p_i}^t$ as well as the hidden scene representation h_c^t estimated by the scene RNN, we now model the relations of the detected persons to infer the activities for each of them. To this end, we represent each detected person as a node ($v_i \in V$) of a fully connected graph. This means that we consider all possible relations how an action of a person effects the actions of the other persons.

As in (Scarselli et al., 2009; Gori et al., 2005; Li et al., 2016), we use a graph RNN that iteratively updates the hidden representation for each node v_i based on the intermediate representations of the other nodes. In our case, the equations for the graph RNN are given by

$$\begin{aligned} x_{v_i}^{(j)} &= maxpool_{v_i \in V}(h_{v_i}^{(j-1)}) \\ a_{v_i}^{(j)} &= x_{p_i} \oplus h_c \oplus x_{v_i}^{(j)} \\ z_{v_i}^{(j)} &= \sigma(U^z a_{v_i}^{(j)} + W^z h_{v_i}^{(j-1)}) \\ r_{v_i}^{(j)} &= \sigma(U^r a_{v_i}^{(j)} + W^r h_{v_i}{}^{(j-1)}) \\ s &= tanh(U^s a_{v_i}^{(j)} + W^s (h_{v_i}^{(j-1)} \circ r_{v_i}^{(j)})) \\ h_{v_i}^{(j)} &= (1 - z_{v_i}^{(j)}) \circ h_v^{(j-1)} + z_{v_i}^{(j)} \circ s \quad (5.4) \end{aligned}$$

where we omitted the frame index t for the ease of reading and $\circ$ denotes the Hadamard product. At each iteration j, the hidden representation for a detected person is given by $h_{v_i}^{(j)}$. To update the representation, we first maxpool the hidden representation over all nodes and concatenate it with the original person feature x_{p_i} as well as the temporal scene context h_c estimated by the scene RNN, which provides a longer temporal context for the graph RNN than the person features x_{p_i}. Using a GRU variant, $h_{v_i}^{(j)}$ is then updated. After a fix number of iterations, the estimated hidden representation for each detected person $h_{v_i}^{(j)}$ is then fed to a fully connected layer with sigmoid as activation function to infer all action classes that are simultaneously performed.

The entire hierarchical graph RNN consisting of the scene RNN and the graph RNN is trained jointly using the focal loss (Lin et al., 2017) for multi-label classifi-

Temporal context l	**0**	**1**	**3**
mAP	19.0%	19.5%	20.9%

Table 5.1: Impact of the temporal context l.

cation.

5.2.3 Implementation Details

The person detector is initialized by a ResNet-101 architecture trained on ImageNet. We finetune the person detector using Adam optimizer with a variable learning rate starting from 0.00001 and dropping by 0.5 at regular intervals. This fine-tuning is done for 100K steps with an effective batch size of 20. The I3D network is pre-trained on Kinetics (Carreira and Zisserman, 2017). Due to memory reasons, we reduce the size of the I3D features from 1024 dimensions to 256 dimensions using a fully connected layer on top of the I3D network and finetune the network on the dataset. For finetuning, we use Adam optimizer with a variable learning rate starting from 0.0005 and dropping by 0.5 at various intervals. This fine-tuning is done for 50K steps with an effective batch size of 50. To make our model robust, we perform data augmentation using random flips and crops as in (Gu et al., 2018).

The hierarchical graph RNN is randomly initialized and trained from scratch using 1000 steps with a batch size of 50 using Adam optimizer with a constant learning rate of 0.0001. Furthermore, we use the focal loss (Lin et al., 2017) with $\gamma = 2$. We train the network on the annotated ground-truth bounding boxes. For inference, we use the detected bounding boxes and we perform a simple multiplication of the person detection confidence with the corresponding action prediction score to obtain the final class predictions for each detection. Finally, class specific non-maximum suppression is used to remove duplicate detections.

5.3 Experiments

For evaluation, we use the AVA 2.1 dataset (Gu et al., 2018). It contains 60 action classes across 235 videos of 15 minutes each for training and 64 videos of the same length for the validation set, which we use for evaluation. Sparse annotations in form of action labels and bounding boxes are provided for a single frame every second. The evaluation is performed using frame-level mean average precision (frame-AP) at IoU threshold 0.5, as described in (Gu et al., 2018).

Temporal Context: To analyze the effect of increasing the temporal support, we evaluate various values for the temporal context l (5.1). Since the frames are sparsely sampled, $l = 3$ corresponds to a temporal context of 7 seconds while $l = 0$ corresponds to 1 second. For the experiment, we use only RGB data without optical flow. The results in Table 5.1 show that increasing the temporal context increases

Models	Graph-RNN	Scene-RNN	HGRNNs
mAP	19.0%	19.8%	20.9%

Table 5.2: Comparison of the Scene-RNN and Graph-RNN with the HGRNN.

Iterations	1	2	3
mAP	20.6%	20.9%	20.8%

Table 5.3: Impact of the number of iterations.

the accuracy. This is due to the fact that certain actions such as 'open' and 'close' can be better recognized with a larger temporal receptive field.

Joint Temporal and Interaction Modeling: In Table 5.1, the accuracy for $l = 0$ corresponds to the case where only the Graph-RNN but not the Scene-RNN is used. In Table 5.2, we also report the accuracy if we use only the Scene-RNN but not the Graph-RNN. In both cases, the proposed Hierarchical Graph-RNN, which combines both RNNs in a single model, achieves a higher accuracy. This shows that both the temporal context as well as the interactions between the persons contribute to the action detection accuracy.

Number of Iterations: As discussed in Section 5.2.2.2, the HGRNN block is iterated. The results in Table 5.3 show that not many iterations are required. This can be attributed to the fact that our Graph-RNN already incorporates temporal information through the hierarchy and it requires only two iterations to update the hidden state of each person based on the hidden states of the other persons. In all other experiments, we use 2 iterations.

Ground Truth (GT) Bounding Boxes: In order to understand the effect of the accuracy of the person detector on the action detection accuracy, we used ground truth bounding boxes during inference. The fine-tuned Faster RCNN person detector achieves an mAP of 89.09% for detecting the annotated bounding boxes on AVA. If ground truth detections are used instead, the action detection accuracy increases by $5 - 6\%$ as shown in Table 5.4. We also report the results if we use RGB and optical flow for computing the I3D features. The additional optical flow increases the accuracy by 2.7% and 3.9% for detected and GT bounding boxes, respectively.

Comparison with State of the Art: The proposed approach outperforms the approaches (Gu et al., 2018; Sun et al., 2018) by a large margin. Most interesting is the comparison to (Gu et al., 2018) since it uses the same features but a vanilla I3D head for action detection. The proposed hierarchical Graph-RNN improves the accuracy by 6.4% on RGB data and 8.0% on RGB+Flow data. This clearly demonstrates the capability of the proposed hierarchical GRNN in comparison to the I3D head (Gu et al., 2018). We also compare our approach with the very recent works (Girdhar et al., 2018; Stroud et al., 2018). While our approach

Method	GT	Detected
RGB	25.2%	20.9%
RGB+Flow	29.1%	23.6%

Table 5.4: Quantitative comparison of the proposed method with ground truth bounding boxes and detected bounding boxes.

Method	flow	mAP
AVA (Gu et al., 2018)		14.5%
ACRN (Sun et al., 2018)		17.4%
Better AVA (Girdhar et al., 2018)		**21.9%**
HGRNN - RGB		20.9%
AVA (Gu et al., 2018)	✓	15.6%
D3D (Stroud et al., 2018)	✓	23.0%
HGRNN - Flow	✓	**23.6%**

Table 5.5: Comparison of the proposed method with other state of the art methods. A ✓at the flow column indicates if optical flow has been used.

outperforms (Stroud et al., 2018), (Girdhar et al., 2018) achieves a higher accuracy for RGB data. The gain of the accuracy is the use of a single Faster RCNN framework that detects the bounding boxes and the action classes together. Using the proposed hierarchical Graph-RNN within a Faster RCNN framework is therefore a future research direction to improve the accuracy further. However, it is unclear how much gain can be achieved if optical flow is used in addition since (Girdhar et al., 2018) does not report any results for optical flow.

5.4 Conclusion

In this chapter, we proposed hierarchical Graph Recurrent Neural Networks for recognizing and localizing multiple activities that occur at the same time. The model learns the temporal context as well as the interactions of the detected persons to recognize the actions. In our experimental evaluation, we have shown that the proposed model outperforms a temporal as well as a graph RNN and that the proposed approach achieves state of the art results on the AVA dataset.

Chapter 6

Discovering Multi-Label Actor-Action Association in a Weakly Supervised Setting

In the previous chapter, we introduced a fully supervised multi-label action detection method that requires detailed annotation for each person. This annotation is costly and time-consuming to obtain. As such, in this chapter, we presented a method for weakly supervised action detection. Our approach focuses on solving actor-action association using linear programming to assign each actor with multiple actions. We build on the top of the HGRNN architecture introduced in the previous chapter to obtain SOTA results on weakly-supervised action detection.

Individual Contribution

The following chapter is based on the publication (Biswas and Gall, 2020):

Discovering Multi-label Actor-Action Association in a Weakly Supervised Setting
Sovan Biswas, and Jürgen Gall.
Asian Conference on Computer Vision (ACCV), 2020.

This publication presents the algorithm and experiments for trimmed videos developed by Sovan Biswas. Jürgen Gall provided scientific guidance and supported this work with very valuable feedback and suggestions.

The idea of temporal linear programming and the additional experiments on untrimmed videos presented in this chapter are not yet published.

Contents

6.1 Introduction

In recent years, we have seen a major progress for spatially and temporally detecting actions in videos (Gkioxari and Malik, 2015; Hou et al., 2017; Kalogeiton et al., 2017; Singh et al., 2017; Sun et al., 2018, 2019; Girdhar et al., 2019; Wu et al., 2019; Feichtenhofer et al., 2019). For this task, the bounding box of each person and their corresponding action labels need to be estimated for each frame as shown in Figure 6.1. Such approaches, however, require the same type of dense annotations for training. Thus, collecting and annotating datasets for spatio-temporal action detection becomes very expensive.

To alleviate this problem, weakly supervised approaches have been proposed (Mettes et al., 2017; Soomro and Shah, 2017; Chéron et al., 2018) where the bounding boxes are not given, but only the action that occurs in a video clip. Despite the promising results of the weakly supervised approaches for spatio-temporal action detection, current approaches are limited to video clips that predominantly contain a single actor performing a single action as in the datasets UCF 101 (Soomro et al., 2012) and JHMDB (Jhuang et al., 2013). However, most real world videos are more complex and contain multiple actors performing multiple actions simultaneously. In this chapter, we move a step forward and introduce the task of weakly supervised multi-label spatio-temporal action detection with multiple actors in a video. The goal is to infer a list of multiple actions for each actor in a given video clip as in the fully supervised case (Sun et al., 2018, 2019; Girdhar et al., 2019; Wu et al., 2019; Feichtenhofer et al., 2019). However, in the weakly supervised setting only actions occurring in each training video are known. Any spatio-temporal information about the persons performing these actions is not provided. This is illustrated in Figure 6.1

Figure 6.1: The image shows a scene where two persons are talking. In this case there are two person that perform multiple actions at the same time. **Person A** indicated by the blue bounding box performs the actions *Stand*, *Listen to*, and *Watch*. **Person B** indicated by the orange bounding box performs the actions *Stand*, *Talk to*, and *Watch*. While in the supervised setting this information is also given for training, we study for the first time a weakly supervised setting where the video clip is only annotated by the actions *Stand*, *Listen to*, *Talk to*, and *Watch* without any bounding boxes or associations to the present persons.

that shows two people standing and chatting. The video clip is only annotated by the four occurring actions *Stand*, *Listen to*, *Talk to*, and *Watch*. Additional information, like bounding boxes or the number of present persons, is not provided. In contrast to previous experimental settings for weakly supervised learning (Mettes et al., 2017; Soomro and Shah, 2017; Chéron et al., 2018), the proposed task is much more challenging since a video clip can contain multiple persons, each person can perform multiple actions at the same time, as well as multiple persons can perform the same action. For instance, both persons in Figure 6.1 perform the actions *Stand* and *Watch* at the same time.

Traditionally, weakly supervised methods (Mettes et al., 2017; Chéron et al., 2018) temporally localize actions by harnessing the fact that two neighboring video frames are likely to have the same action labels. This is performed by forming tubelets that are linked using a greedy algorithm or dynamic programming over mutually exclusive action scores. In a multi-label setting, actions are also likely to be similar in the temporal neighborhood. However, the temporal extent of each action performed by the same actor can vary as shown in Figure 6.2. For example, person A continues to "*listen*" and "*stand*" during the entire video clip but performs "*watch*" towards the end of the clip. This restricts the usage of traditional greedy algorithms or dynamic programming over a set of co-occurring actions. Nevertheless, the temporal similarity of actions is still a strong clue for action localization in a multi-label setting.

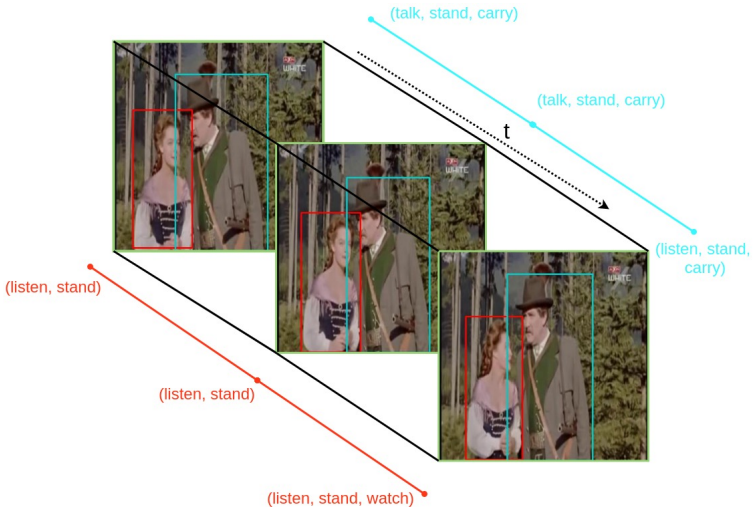


Figure 6.2: The video clip of time t shows a scene where two persons are chatting. **Person A** indicated by the red bounding box performs the actions *Stand*, *Listen* and *Watch* over time. **Person B** in cyan bounding box performs the actions *Stand*, *Talk*, *Carry* and *Listen*. The weak video clip annotation only contains the list of actions *Stand*, *Listen*, *Talk*, *Carry* and *Watch* without any bounding boxes or any annotation for person re-identification.

In order to address multi-label spatio-temporal action detection in the proposed weakly supervised setup, we first introduce a baseline that uses multi-instance and multi-label (MIML) learning (Zhou and Zhang, 2006; Zhou et al., 2012; Yang et al., 2017). Second, we introduce a novel approach that is better suited for the multi-label setting. Instead of modeling the class probabilities for each action class, we build the power set of all possible action combinations and model the probability for each subset of actions. Using a set representation has the advantage that we model directly the combination of multiple occurring actions instead of the probabilities of single actions. Since computing the probabilities for the full power set becomes intractable as the number of action classes increases, we assign an action set to each detected person under the constraint that the assignment is consistent with the annotation of the video clip. This is done by linear programming, which maximizes the overall gain across all plausible actors and action subset combinations. Lastly, we incorporate a soft temporal consistency measure in the proposed framework that ensures temporal consistent actor-action assignments. This consistent assignment is of higher importance for longer clips as seen in improvements in the performance for those clips. Note, the proposed soft temporal consistency measure considers that an actor can transits from one action to another as shown in Figure 6.2.

We evaluate the proposed approach on the challenging AVA 2.2 dataset (Gu et al., 2018), which is currently the only dataset that can be used for evaluating

this task. In our experiments, we show that the proposed approach outperforms the MIML baseline by a large margin and that the proposed approach achieves 83% of the mAP compared to a model (on same backbone) trained with full supervision.

In summary, the contribution of this approach is three-fold:

- We introduce the novel task of weakly supervised multi-label spatio-temporal action detection with multiple actors.
- We introduce a first baseline for this task based on multi-instance and multi-label learning.
- We propose a novel approach based on an action set representation.
- We explore temporal label consistency to prune noisy action set representation.

6.2 Multi-instance and Multi-label (MIML) Learning for Action Detection and Recognition

Given a video clip with multiple actors where each actor can perform multiple actions at the same time as shown in Figure 6.1 and Figure 6.2, the goal is to localize these actors and predict for each actor the corresponding actions. In contrast to fully supervised learning (Gkioxari and Malik, 2015; Sun et al., 2019; Girdhar et al., 2019; Wu et al., 2019; Pan et al., 2021), where bounding boxes with multiple action labels are given for training, we address for the first time a weakly supervised setting where only a list of actions is provided for each video clip during training. This is a very challenging task as we do not know how many actors are present and each actor can perform multiple actions at the same time. This is in contrast to previous weakly supervised spatio-temporal action localization (Mettes et al., 2017; Soomro and Shah, 2017; Chéron et al., 2018) where it is assumed that only one person is in the video and that the person does not perform more than one action at a given point in time. Further, this weakly supervised spatio-temporal action localization presents additional challenges such as temporal consistency of actions, memory limitations, etc. for long untrimmed video.

One way to address the weakly supervised learning problem is to use multiple-instance learning. Since we have a multi-label problem, i.e., an actor can perform multiple actions at the same time, we use the concept of multi-instance and multi-label (MIML) learning (Zhou and Zhang, 2006; Zhou et al., 2012; Yang et al., 2017). We first use a person detector (Xie et al., 2017) to spatially localize the actors in a frame t and use a 3D-CNN such as I3D (Carreira and Zisserman, 2017) or Slowfast (Feichtenhofer et al., 2019) for predicting the action probabilities similar to fully supervised methods (Girdhar et al., 2019; Wu et al., 2019). However, we use the MIML loss to train the networks.

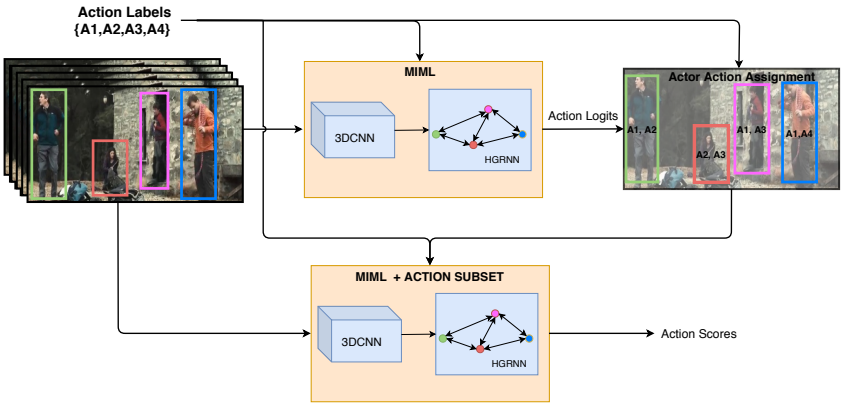


Figure 6.3: Overview of the proposed approach. Given a training video clip with action labels {A1,A2,A3,A4}, we first detect persons in the video. We then train a 3D CNN with a graph RNN that models the spatio-temporal relations between the detected persons using the MIML loss to obtain initial estimates of the action logits. During actor-action association, subsets of the action labels are assigned to each detected person. The training of the network is continued using the MIML loss and the actor-action associations.

We denote by $A_t = \{a_1^t, a_2^t, \ldots, a_{n_t}^t\}$ the detected bounding boxes and by $f(a_i^t)$ the class probabilities that are predicted by the 3D-CNN. Let Y be the vector which contains the annotations of the video clip, i.e., $Y(c) = 1$ if the action class c occurs in the video clip and $Y(c) = 0$ otherwise. In other words, the bag A_t is labeled by $Y(c) = 1$ if at least one actor performs the action c and by $Y(c) = 0$ if none of the actors performs the action. The MIML loss is then given by

$$\mathcal{L}_{miml} = \mathcal{L}\left(Y, \max_i(f(a_i^t))\right) \tag{6.1}$$

$$\mathcal{L}_{un} = \frac{1}{\sigma^2}\mathcal{L}\left(Y, \max_i(f(a_i^t))\right) + log\sigma^2 \tag{6.2}$$

where $\mathcal{L}$ is the standard binary cross entropy loss function. The standard MIML ensures that the class probability should be close to one for at least one bounding box if the action is present and it should be close to zero for all bounding boxes if the action class is not present as shown in (6.1). On the other hand, uncertainty aware MIML (Arnab et al., 2020) (6.2) re-scales the loss function based on the uncertainty of the prediction. σ is the uncertainty of the prediction.

6.3 Actor-Action Association: Trimmed Videos

While multi-instance and multi-label learning discussed in Section 6.2 already provides a good solution for the new task of weakly supervised multi-label action detection, we propose in this section a novel method that outperforms the results by a large margin. We first change the representation from individual action labels to sets of actions by building power set of all possible action combinations. The objective is then to assign each person with a set of actions such that each person has at least one action and all the actions of the annotation exists. For example, the power set Ω for the three action labels *Listen*, *Talk*, and *Watch* is $\{\varnothing, \{Listen\}, \{Talk\}, \{Watch\}, \{Listen, Talk\}, \{Listen, Watch\}, \{Talk, Watch\}, \{Listen, Talk, Watch\}\}$. With that formulation in mind, then we assign one set $\omega_i^t \in \Omega \setminus \varnothing$ to each actor a_i^t under the constraint that each action c occurs at least once, i.e., $c \in \bigcup_i^t \omega_i^t$. Using a set representation has the advantage that we model directly the combination of multiple occurring actions instead of the probabilities of single actions. This means that we have one probability for a subset of actions $\omega \in \Omega$ instead of C probabilities where C is the number of action labels. We propose a novel method to compute the probability of a set actions in Section 6.3.1. Due to the weakly supervised setting not all combinations of subsets are possible for each video clip. We therefore assign an action set $\omega_i^t \in \Omega$ to each actor a_i^t under the constraint that the assignment is consistent with the annotation of the trimmed video clip, i.e., each annotated action c needs to occur at least once and actions that are not annotated should not occur. The assignment is discussed in Section 6.3.2.

Figure 6.3 illustrates the complete approach. As described in Section 6.2, we use a 3D CNN such as I3D (Carreira and Zisserman, 2017) or Slowfast (Feichtenhofer et al., 2019) as backbone. Since the actors in a frame often interact with each other, we use a graph to model the relations between the actors. The graph connects all actors and we use a graph RNN to infer the action probabilities for each actor based on the spatial and temporal context. In our approach, we use the hierarchical Graph RNN (HGRNN) (Chapter 5) where the features per node are obtained by ROI pooling over the 3D CNN feature maps. The HGRNN and 3D CNN are learned using the MIML loss (6.1). From the action class probabilities, we infer the action set probabilities as described in Section 6.3.1 and we infer the action set for each actor as described in Section 6.3.2. Finally, we train the HGRNN and the 3D CNN based on the assignments. This will be discussed in Section 6.5.

6.3.1 Power Set of Actions

In principle, we could modify our network to predict the probability for each subset of all action classes instead of the probabilities for all action classes. However, this is infeasible since the power set of all actions is very large. If C is the number of actions in a dataset, the power set for all actions consists of 2^C subsets. Already with 50

action classes, we would need to predict the probabilities for over one quadrillion subsets. Instead, we use an idea that was proposed for HEX graphs (Deng et al., 2014) where the probabilities of a hierarchy are computed from the probabilities of the leave nodes. While we do not use a hierarchy, we can approximate the probability of a subset of actions from the predictions of a network for individual actions.

Let $s_c \in (-\infty, \infty)$ denote the logit that is predicted by the network for the action class c. The probability of a subset of actions ω can then be computed by

$$p_\omega = \frac{\exp\left(\sum_{c\in\omega} s_c\right)}{\sum_{\omega'} \exp\left(\sum_{c\in\omega'} s_c\right)}. \tag{6.3}$$

The normalization term, however, is still infeasible to compute since we still need to sum over all possible subsets (ω') for the dataset.

Since our goal is the assignment of a subset of actions ω to each actor, we do not need to compute the full probability (6.3). Instead of using the power set of all actions, we build the power set only for the actions that are provided as weak labels for each training video clip. This means that the power set will differ for each video clip. For the example shown in Figure 6.1, we build the power set Ω for the actions *Stand*, *Listen*, *Talk*, and *Watch*. In this example, $|\Omega| = 16$. We exclude $\varnothing$ since in the used dataset each actor is annotated with at least one action. Furthermore, we multiply p_ω with the confidence d of the person detector. The scoring function $p_{\omega,i}$ that we use for the assignment of a subset $\omega \in \Omega \setminus \varnothing$ to a detected actor a_i is therefore given by

$$p_{\omega,i} = \frac{\exp\left(\sum_{c\in\omega} s_{c,i}\right) d_i}{\sum_{\omega'\in\Omega\setminus\varnothing} \exp\left(\sum_{c\in\omega'} s_{c,i}\right)} \tag{6.4}$$

where $s_{c,i}$ is the predicted logit for action c and person a_i. Taking the detection confidence d_i of person a_i into account is necessary to reduce the impact of false positives that usually have a low detection confidence.

6.3.2 Actor-Action Association

While the scoring function (6.4) indicates how likely a given subset of actions $\omega \in \Omega\setminus\varnothing$ fits to an actor a_i, it does not take all information that is available for each video clip into account. For instance, we know that each annotated action is performed by at least one actor. In order to exploit this additional knowledge, we find the optimal assignment of action subsets to actors based on the constraints that each actor performs at least one action and that each action c occurs at least once, i.e., $c \in \bigcup_i \omega_i$. Since we build the power set only from the actions that occur in a video clip, which we denote by L, the power set $\Omega(L)$ varies for each training video clip.

The association of subsets $\omega \in \Omega(L) \setminus \varnothing$ to actors $A = \{a_1, a_2, \ldots, a_n\}$ can be formulated as a binary linear program where the binary variables $x_{\omega,i}$ are one if the subset ω is assigned to actor a_i and it is zero otherwise. The optimal assignment is de-

fined by the assignment with the highest score (6.5). While the first constraint (6.6) enforces that exactly one subset ω is assigned to each actor a_i, the second constraint (6.7) enforces that $c \in \bigcup_{\omega:x_{\omega,i}=1} \omega$ for all $c \in L$, where $\{\omega : x_{\omega,i} = 1\}$ is the set of all subsets that have been assigned. Note that (6.7) rephrases this constraint such that it can be used for optimization where the indicator function $\mathbb{1}_\omega(c)$ is one if $c \in \omega$ and it is zero otherwise. The left hand side of the inequality therefore counts the number of assigned subsets that contain the action class c. Since this number must be larger than zero, it ensures that each action $c \in L$ is assigned to at least one actor. The complete binary linear program is thus given by:

$$\underset{x_{\omega,i}}{\arg\max} \sum_{i=1}^{n} \sum_{\omega \in \Omega(L)\setminus\varnothing} p_{\omega,i} x_{\omega,i} \tag{6.5}$$

$$\text{subject to} \sum_{\omega \in \Omega(L)\setminus\varnothing} x_{\omega,i} = 1 \qquad \forall i = 1, ..., n \tag{6.6}$$

$$\sum_{i=1}^{n} \sum_{\omega \in \Omega(L)\setminus\varnothing} \mathbb{1}_\omega(c) x_{\omega,i} \geq 1 \qquad \forall c \in L \tag{6.7}$$

$$x_{\omega,i} \in \{0, 1\} \qquad \forall \omega \in \Omega(L) \setminus \varnothing;\ \forall i = 1, ..., n.$$

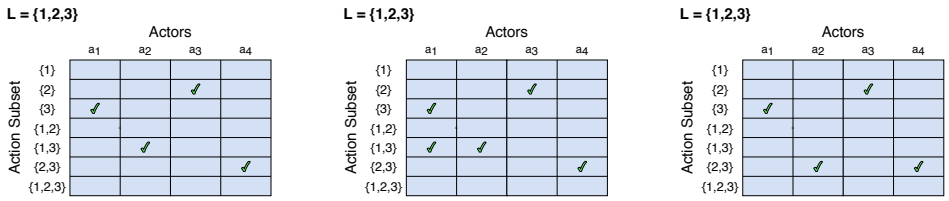
Figure 6.4 illustrates the constraints.

6.4 Actor-Action Association: Untrimmed videos

Even though the iterative approach of actor-action association based on linear programming is suitable for trimmed videos, the same formulation does not work for untrimmed videos due to possibility of action transitions as shown in Figure 6.2. Moreover, the assumption that each detected person must be associated with an action is flawed, as person detections can include false positives or some individuals may be engaged in irrelevant background actions in video. Therefore, we propose modifications to the previously proposed linear programming (Section 6.3.2) to handle long untrimmed videos. We also replace the Hex-graph based subset scores (Section 6.3.1) to a novel assorted scores that allows for null-set ($\varnothing$) assignments and integrates temporal consistency to ensure stable action labels across time.

6.4.1 Temporal Linear Programming

Similar to trimmed videos, each untrimmed video clip is annotated by a set of weak action labels (L) as shown in Figure 6.2. The powerset of labels $\Omega(L)$ provides all possible subset actions for each actor $a_{i,t}$ at any given time t. The objective of the actor-association is still to associate one of such subset $\omega \in \Omega(L)$ to each actor $a_{i,t}$ based on its subset score. However, in contrast to trimmed videos, untrimmed



a) Valid assignment. b) Invalid assignment. c) Invalid assignment.

Figure 6.4: For the annotated actions $L = \{1, 2, 3\}$ and the actors $A = \{a_1, a_2, a_3, a_4\}$, the figures demonstrate various actor-action assignments. While the assignment a) satisfies all constraints, b) violates (6.6) since two subsets are assigned to actor a_1 and c) violates (6.7) since the action 1 is not part of any assigned subset.

videos can contain action transitions. Yet, most of the actions in action subset for an actor is relatively consistent over time (as seen in Figure 6.2). To incorporate the ability to handle action transitions, we divide the untrimmed videos into T equal-sized trimmed clips. We assume that actions remain consistent within each trimmed clip, and that transitions of one action to another can only occur between two adjacent clips. Using this premise, we modify the above linear programming, to find the optimal assignment of actions to each actor based on the similar constraints of Section 6.3.2, i.e. each actor performs at least one action and that each action c occurs at least once in whole untrimmed video clip ($c \in \bigcup_{i,t} \omega_{i,t}$). We build the power set only from the actions based on the weak annotations (L). The power set $\Omega(L)$ varies for each untrimmed video clip. The association of subsets $\omega \in \Omega(L)$ to actors $A = \{A_1, A_2, \ldots, A_T\}$ (where $A_t = \{a_{1,t}, a_{2,t}, \ldots, a_{n,t}\}$) can be formulated into a binary linear program as:

$$\underset{x_{\omega,i,t}}{\arg\max} \sum_{\omega \in \Omega(L)} \sum_{t=1}^{T} \sum_{i=1}^{|A_t|} \alpha_{\omega,i,t} x_{\omega,i,t} \tag{6.8}$$

$$\text{s.t.} \sum_{\omega \in \Omega(C)} x_{\omega,i,t} = 1 \qquad \begin{matrix} \forall i = 1, ..., |A_t| \\ \forall t = 1, ..., T \end{matrix} \tag{6.9}$$

$$\sum_{\omega \in \Omega(L)} \sum_{t=1}^{T} \sum_{i=1}^{|A_t|} \mathbb{1}_{\omega}(c) x_{\omega,i,t} \geq 1 \qquad \forall c \in L \tag{6.10}$$

$$x_{\omega,i,t} \in \{0, 1\} \qquad \begin{matrix} \forall \omega \in \Omega(L) \\ \forall i = 1, ..., |A_t| \\ \forall t = 1, ..., T \end{matrix}$$

Here, the non-negative $\alpha_{\omega,i,t}$ is the assorted subset score (explained in detail in

Section 6.4.2) and $x_{\omega,i,t}$ is the binary variable whose value is 1 if the subset ω is assigned to actor $a_{i,t}$. Note, the linear program along with its constraints, is applied to the entire untrimmed video that was divided into T trimmed clips. This setup enables action transitions between consecutive clips. However, because we have not used actor tracking, the optimal actor-action assignments in successive clips may exhibit temporal inconsistencies. Thus, we incorporate temporal consistency score within $\alpha_{\omega,i,t}$ to prevent such inconsistent label assignments.

6.4.2 Assorted Subset Scores

The effectiveness of this actor-action association using the linear program heavily relies on the scoring ($\alpha_{\omega,i,t}$) for each possible subset for the i^{th} actor at time t. These scores must not only accurately represent each subset but also favour temporal consistency. Moreover, even highly accurate person detectors can produce false positives, so the non-negative subset scores should be low for such cases. To achieve this, we proposed an assorted subset score $\alpha_{\omega,i,t}$ that combines three terms to represent each subset ω:

$$\alpha_{\omega,i,t} = p_{\omega,i,t} + \tau_{\omega,i,t} + r_{\omega,i,t}. \tag{6.11}$$

It includes a subset score ($p_{\omega,i,t}$) computed from the probabilities of individual actions, a temporal consistency score ($\tau_{\omega,i,t}$) to ensure consistent action assignment for each actor over time, and a re-scaled subset score ($r_{\omega,i,t}$) to boost the low action probabilities for the tail classes.

Subset score ($p_{\omega,i,t}$)

We proposed to compute the score for a subset $\omega \in \Omega(L)$ for a detected actor $a_{i,t}$, using initial action probabilities $p_{c,i,t}$ and the detection confidences $d_{i,t}$. $p_{c,i,t}$ is the probability score for c^{th} class from the model for i^{th} detection at time t. L is the set of action labels for the untrimmed video clip. The score for any given action subset ω is computed as

$$p_{\omega,i,t} = \begin{cases} \prod_{c \in L}(1 - (p_{c,i,t} * d_{i,t})) & \text{if } \omega = \varnothing \\ d_{i,t} * \prod_{c \in \omega} p_{c,i,t} & \text{otherwise} \end{cases} \tag{6.12}$$

This scoring function has the advantage that it not only takes the detection confidences into account but also provides a score for $\omega = \varnothing$. Indeed, if the person detection confidence is very low or if the action scores $p_{c,i,t}$ are low for all $c \in L$, then it is likely to be a false positive. In such case, $p_{\omega,i,t}$ is low for all non empty subsets and close to one for the null-set ($\omega = \varnothing$). Furthermore, by multiplying the individual class probabilities, the scoring function is designed to encourage smaller sized subset ω to have high score. This avoids a trivial solution where all the actions

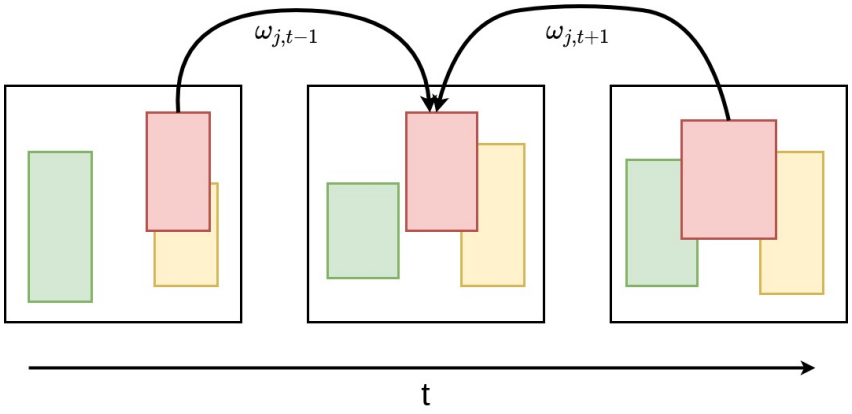


Figure 6.5: After initial estimation of the action labels for each actor proposal, we use the neighboring frames to update the labels based on temporal consistency.

labels (L) are assigned to a single person and rest persons are assigned null-set i.e. $\omega = \varnothing$.

Temporal Consistency Score($\tau_{\omega,i,t}$)

Actions performed by an actor often span multiple frames as shown in Figure 6.2, where both individuals are standing throughout the entire clip. To obtain this temporally consistent behavior, we begin by solving the above linear programming without temporal consistency score to obtain an initial label assignment for each person in the entire untrimmed clip. Then, for all proposals A_t at time t, we consider the action subset assignment of the neighboring frames $t-1$ and $t+1$ into account as shown in Figure 6.5, to compute the temporal consistency score (6.13) and update the assorted scores $\alpha_{\omega,i,t}$. Consequently, the computation of the temporal consistency score is an iterative process, with each score being updated based on label assignments from the previous iteration.

The idea is that the score $\tau_{\omega,i,t}$ for a_i is high for a subset ω when proposals a_j in neighboring frames are spatially close to a_i and share a similar assignment to ω.

$$\begin{aligned}\tau_{\omega,i,t} = &\sum_{j=1}^{|A_{t-1}|} I(a_{i,t}, a_{j,t-1}) S(\omega, \omega_{j,t-1}) B(\tilde{\alpha}_{\omega,j,t-1}) \\ &+ \sum_{j=1}^{|A_{t+1}|} I(a_{i,t}, a_{j,t+1}) S(\omega, \omega_{j,t+1}) B(\tilde{\alpha}_{\omega,j,t+1}) \end{aligned} \tag{6.13}$$

where

$$\begin{aligned} I(a_i, a_j) &= IOU(a_i, a_j) \\ S_{\omega_i,\omega_j} &= \exp\left(-2h(\omega_i, \omega_j)\right) \\ B(\tilde{\alpha}_{\omega,i,t}) &= \frac{\tilde{\alpha}_{\omega,i,t}}{\max_\omega \tilde{\alpha}_{\omega,i,t}}. \end{aligned}$$

The spatial similarity $I(a_i, a_j)$ is measured by the intersection-over-union of the bounding boxes of the persons. The similarity S_{ω_i,ω_j} of two action sets is measured by the Hamming distance h. $B(\tilde{\alpha}_{\omega,i,t})$ denotes the confidence of the previous assignment where $\tilde{\alpha}_{\omega,j,t-1}$ denotes the assorted score. In this way, the temporal consistency score favours action subsets that are consistent with its temporal neighbourhood.

Re-scaled Subset Score ($r_{\omega,i,t}$)

If a video clip contains difficult examples for an action class (c), the highest recognition probability for that action over all the proposals can be very low (< 0.5). Due to the constraints and subset scoring function (6.12), solving the linear program with such low score can produce trivial solutions. To tackle the issue, we perform class-wise re-scaling of each action score ($p_{c,i,t}$) with respect to maximum score ($m_c \in (0, 1)$) of the untrimmed clip as shown in (6.14). This boosts the confidence of difficult examples of an action class independent of the confidence of other action classes. Mathematically,

$$\begin{aligned} m_c &= \max_{i,t} p_{c,i,t} \\ r_{c,i,t} &= \frac{p_{c,i,t}}{m_c} \end{aligned} \tag{6.14}$$

where, $p_{c,i,t}$ denotes the c^{th} action score for i^{th} proposal at time t. m_c is the maximum score of action c in the untrimmed clip. This rescaled action score ($r_{c,i,t}$) replaces $p_{c,i,t}$ in (6.12) to compute re-scaled subset score ($r_{\omega,i,t}$).

6.5 Training

6.5.1 Negatives Mining

Due to weak training annotations, MIML is prone to over-fitting the training data. To address this issue and enhance robustness during training, we use negative mining. For each video clip, we randomly generate b_k proposals such that the IOU of these new proposals with actor-proposals a_i is less than 0.2. In other words, these generated proposals do not overlap with any actor in the video and serve as negative examples for every action, providing additional supervision during training. Training with such negatives not only strengthens model robustness but also improves

the model's ability to identify false positives among actor proposals. We have set b_k to the batch size.

6.5.2 Loss Functions

The proposed approach uses an iterative strategy that alternates between model training and actor-action assignment multiple times. In our first iteration, we train the model using the MIML loss (6.1) or uncertainty-aware MIML loss (6.2) with negative mining (Section 6.5.1) to account for the lack of any detailed annotation to the proposals. This is done to obtain initial estimates of the action probabilities $p_{c,i,t}$. We then assign subsets of actions to the detected persons using the scoring function (6.4) and (6.11) for trimmed and untrimmed videos respectively. In the subsequent iterations, we train our model using the actor action associations, the loss is denoted by $\mathcal{L}_{aaa}$ and $\mathcal{L}_{aaa+}$ for trimmed and untrimmed videos respectively, as shown below:

$$\mathcal{L}_{miml} = \mathcal{L}\left(Y, \max_i(f(a_i^t))\right)$$

$$\mathcal{L}_{un} = \frac{1}{\sigma^2}\mathcal{L}\left(Y, \max_i(f(a_i^t))\right) + log\sigma^2$$

$$\mathcal{L}_{neg} = \mathcal{L}\left(Y_{\varnothing}, f(b_k^t))\right)$$

$$\mathcal{L}_{pos} = \mathcal{L}\left(\hat{Y}_{\omega_i^t}, f(a_i^t)\right)$$

$$\mathcal{L}_{aaa} = \mathcal{L}_{miml} + \eta\mathcal{L}_{pos} \tag{6.15}$$

$$\mathcal{L}_{aaa+} = \mathcal{L}_{un} + \mathcal{L}_{neg} + \eta\mathcal{L}_{pos} \tag{6.16}$$

where, $\mathcal{L}$ is binary cross entropy loss. σ denotes the uncertainty in the prediction (Arnab et al., 2020). a_i^t is the i^{th} actor-proposal and b_k^t is the k^{th} negative proposal at time t of the video clip. Y is list of actions (C) represented as binary vector with $Y(c) = 1$ if $c \in C$ and $Y(c) = 0$ otherwise. $Y_{\varnothing}$ is a vector with $Y_{\varnothing}(c) = 0$ $\forall c \in \mathbb{U}$. ω_i^t denotes the action subset that has been assigned to actor a_i^t in frame t and $\hat{Y}_{\omega_i^t}$ is a vector with $\hat{Y}_{\omega_i^t}(c) = 1$ if $c \in \omega_i^t$ and $\hat{Y}_{\omega_i^t}(c) = 0$ otherwise. We use $\eta = 0.1$ based on the experiments as shown later.

6.6 Experiments

6.6.1 Dataset and Implementation Details

We use the AVA 2.2 dataset (Gu et al., 2018) for evaluation. The dataset contains 235 videos for training, 64 videos for validation, and 131 videos for testing. The dataset contains 60 action classes. Each individual often performs multiple actions at the same time and the videos contain multiple persons. For each annotated person

Method	3D CNN	Val-mAP
MIML	I3D	14.1
MIML + HGRNN	I3D	15.2
$\mathcal{L}_{aaa}$	I3D	17.3
MIML	SF-101	21.8
MIML + HGRNN	SF-101	23.1
$\mathcal{L}_{aaa}$	SF-101	25.1

Table 6.1: Comparison of MIML with proposed method on trimmed videos. The proposed approach outperforms MIML in case of I3D and Slowfast 101 (SF-101).

a bounding box is provided. An example is given in Figure 6.1. Only one frame per second is annotated. The accuracy is measured by mean average precision (mAP) over all actions with an IoU threshold for bounding boxes of 0.5 as described in (Gu et al., 2018). In the weakly supervised setting, we use only the present actions for training, but not the bounding boxes.

To detect persons, we use Faster RCNN (Ren et al., 2015) with ResNext-101 (Xie et al., 2017) as backbone. The detector was pre-trained on ImageNet and fine-tuned on the COCO dataset. In our experiments, we report results for two 3D CNNs, namely I3D (Carreira and Zisserman, 2017) and Slowfast (Feichtenhofer et al., 2019). I3D is pre-trained on Kinetics-400. For Slowfast, we use the ResNet-50 and ResNet-101 + NL (8×8) version that is pre-trained on Kinetics 600, indicated by SF-50 and SF-101 respectively. The temporal scope was set to 64 frames with a stride of 2. For HGRNN we use a temporal window of 11 frames. For training, we use the SGD optimizer until the validation error saturated. The learning rate with linear warmup was set to 0.04 and 0.025 for I3D and Slowfast (both SF-50 and SF-101), respectively. The batch size was set to 16. We used cropping as data augmentation where we crop images of size 224×224 pixels from the frames that have 256×256 image resolution.[1]

In this dataset, the untrimmed videos can have scene-changes within them. We set the temporal context of the neighbour to 0, if the corresponding neighbouring trimmed video belongs to a different scene. This reduces the impact of erroneous temporal context in lie of scene change. We use PySceneDetect (Castellano, 2018) to perform the off-the-shelf scene change detection.

6.6.2 Actor-Action Assignment on trimmed videos without ∅

Comparison of MIML with proposed method.

Table 6.1 shows the comparison of the proposed approach with the multi-instance and multi-label (MIML) baseline on the validation set. When I3D is used as 3D

[1]Code: https://github.com/sovan-biswas/MultiLabelActorActionAssignment

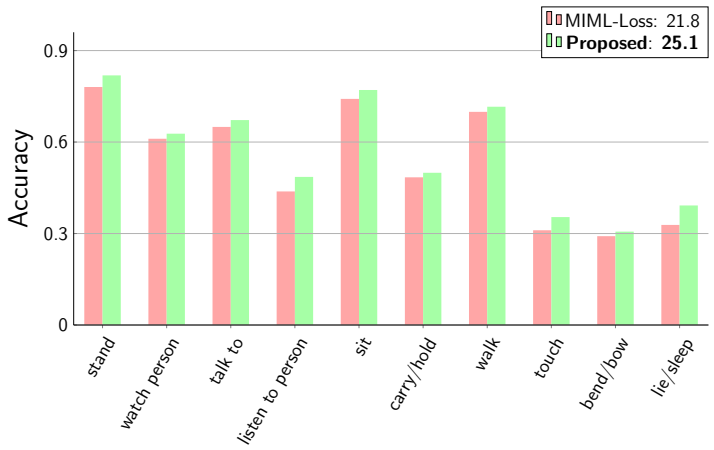


Figure 6.6: Comparison of MIML with proposed method on trimmed videos. The plot shows the per class mAP for the 10 most frequently occurring classes in the training set. The actions are sorted by the number of occurrences in an decreasing order from left to right.

CNN, the proposed approach improves the MIML baseline by +3.2%. When SF-101 is used, the accuracy of all methods is higher but the improvement of the proposed approach over the MIML approach remains nearly the same with +3.3%. We also report the result when HGRNN is trained only with the MIML loss. In this case, the actor-action association is not used and we denote this setting by MIML+HGRNN. While HGRNN improves the results since it models the spatio-temporal relations between persons better than a 3D CNN alone, the proposed actor-action assignment improves the mAP compared to MIML+HGRNN by +2.1% and +2.0% for I3D and SF-101, respectively. Figure 6.6 shows the improvement of the proposed approach over the MIML baseline for the 10 action classes that occur most frequently in the training set. A few qualitative results are show in Figure 6.8.

Impact of Linear programming for actor-action assignment

In Table 6.1, we have observed that the actor-action association improves the accuracy. In Table 6.2, we analyze the impact of the actor-action association more in detail. We use HGRNN using both I3D and Slowfast as 3D CNN backbone. In case of MIML+HGRNN, the actor-action association is not used. We also report the result if we perform the association directly by the confidences without solving a binary linear program. We denote this setting by $\mathcal{L}_{aaa}$ w/o LP. In this case, we associate an action to an actor if the class probability is greater than 0.5. For I3D, the association without LP improves the results mainly for the most frequent classes with almost no improvement on least frequent classes. For SF-101, the performance

Actor-action association	Backbone	Val-mAP	Frequent-5	Least-10
MIML+HGRNN	I3D	15.2	51.5	2.0
$\mathcal{L}_{aaa}$ w/o LP	I3D	16.4	52.8	2.1
$\mathcal{L}_{aaa}$	I3D	**17.3**	**53.7**	**3.4**
MIML+HGRNN	SF-101	23.1	65.7	7.3
$\mathcal{L}_{aaa}$ w/o LP	SF-101	22.9	65.9	6.8
$\mathcal{L}_{aaa}$	SF-101	**25.1**	**67.5**	**7.6**

Table 6.2: Results of various actor-action assignment approaches using HGRNN over different 3D CNNs. The Frequent-5 column and the Least-10 column show the average mAP over the 5 most frequently and 10 least occurring classes in the training set.

Method	3D CNN	Detected bb	GT bb
MIML	I3D	14.1	21.2
$\mathcal{L}_{aaa}$	I3D	17.3	24.3
Full Supervision	I3D	20.7	25.4
MIML	SF-101	21.8	30.6
$\mathcal{L}_{aaa}$	SF-101	25.1	32.3
Full Supervision	SF-101	30.1	35.7

Table 6.3: Performance with ground-truth bounding boxes for evaluation. The results show the improvement in mAP on the validation set when ground-truth bounding boxes (GT bb) instead of detected bounding boxes (Detected bb) are used for evaluation. Furthermore, the results are reported when the model is trained with full supervision.

even decreases in comparison to MIML+HGRNN without LP. Instead, solving the linear program results in better associations for both I3D and Slowfast.

Impact of the object detector

We use the Faster RCNN with ResNext (Xie et al., 2017) person detector which achieves 90.10% mAP for person detection on the AVA training set and 90.45% on the AVA validation set. Irrespective of the high detection performance, we analyze how much the accuracy improves if the detected bounding boxes are replaced with the ground-truth bounding boxes during evaluation. Note that the ground-truth bounding boxes are not used for training, but only for evaluation. The results are shown in Table 6.3. We observe that the performance improves by +7.0% and +7.2% mAP on the validation set for I3D and Slowfast, respectively. We also report the results if the approach is trained using full supervision. In this case, the network is trained on the ground-truth bounding boxes and the ground-truth action labels per bounding box. Compared to the fully supervised approach, our weakly supervised

Method	Backbone	$t = 1$
$\mathcal{L}_{miml}$	SF-101	21.8
$\mathcal{L}_{aaa}$		22.5
$\mathcal{L}_{miml}$	SF-101 + HGRNN	23.1
$\mathcal{L}_{aaa}$		25.1

Table 6.4: Results of various actor-action assignment approaches using SF-101 and HGRNN for label assignment. HGRNN helps in better label assignment.

Method	Backbone	$t = 1$	$t = 5$	$t = 10$
$\mathcal{L}_{miml}$	SF-50	20.8	18.4	17.1
$\mathcal{L}_{miml} + \mathcal{L}_{neg}$		21.8	18.8	17.8
$\mathcal{L}_{aaa+}$		**22.2**	**20.2**	**19.0**
$\mathcal{L}_{miml}$	SF-101	23.1	-	-
$\mathcal{L}_{miml} + \mathcal{L}_{neg}$		24.0	21.1	19.0
$\mathcal{L}_{aaa+}$		**24.8**	**22.1**	**20.2**

Table 6.5: Performance of the proposed approach in comparison to baseline and impact of negative mining. Note: t is the length of the untrimmed videos in seconds.

approach achieves around 83% of the mAP for both 3D CNNs (17.3% vs. 20.7% for I3D and 25.1% vs. 30.1% for Slowfast) if detected bounding boxes are used for evaluation. The gap gets even smaller when ground-truth bounding boxes are used for evaluation. In this case, the relative performance is 95.7% for I3D and 90.5% for Slowfast. This demonstrates that the proposed approach learns the actions very well despite of the weak supervision.

Impact of HGRNN for Actor-Association

In Table 6.4, we analyze the impact of HGRNN in weak supervised action detection. In case of proposed $\mathcal{L}_{aaa}$ w/o HGRNN, actor-action assignment is performed with MIML logits and HGRNN is not used at any stage. The proposed actor-action assignment improves the result in both cases. In case of HGRNN, the improvement is more due to HGRNN's ability to incorporate spatial and temporal information as seen in Table 6.4.

Effect of η

η from (6.15) seems to have minor effect on performance of the proposed approach as seen in Figure 6.7. Here, $\eta = 0$ corresponds to $\mathcal{L}_{miml}$ when actor-action assignment is not used. It is clear that emphasing more on the assignment can be detrimental. This is because of the fact that these assignment contains errors which could lead the model towards poor performance as seen with $\eta = 1$ in Figure 6.7.

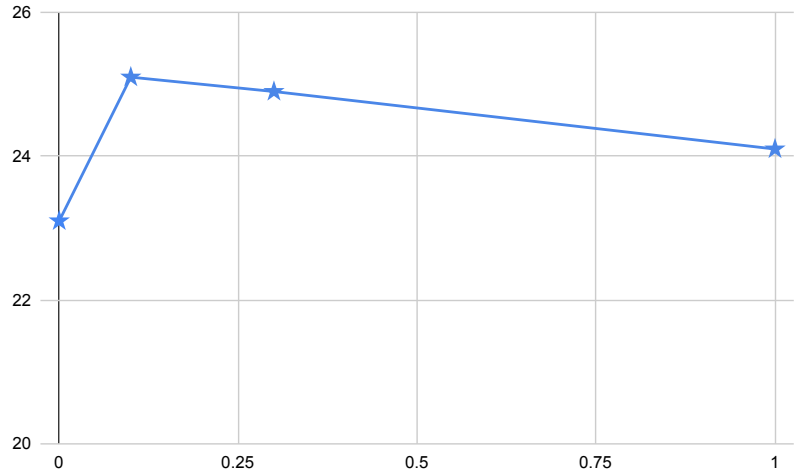


Figure 6.7: Effect of varying η of (6.15)

6.6.3 Actor-Action Assignment on untrimmed videos with null-set ∅

Baseline MIML and negative mining

For our actor-action assignment with null-set ∅, we obtain our initial estimation with miml loss with negative mining. Thus, we first explore the impact of negative mining on the performance as shown in Table 6.5. We start our experiments with standard MIML loss for different backbones and different values of t i.e. different video length. We have not used HGRNN in this experiment. The standard MIML loss ($\mathcal{L}_{miml}$) results in a mAP of 20.8% and 23.1% with SF-50 and SF-101, respectively for trimmed videos (i.e. $t = 1$). The negative mining uses random proposals from non actor locations as negatives for action detection as suggested in Section 6.5.1. This enables our approach to better discriminate between actual person and false positives. As a result, our approach achieves the mAP scores ($\mathcal{L}_{miml} + \mathcal{L}_{neg}$) of 21.8% and 24.0%, which is an improvement of +1.0% and 0.9% respectively than the standard MIML. In fact, the negative mining even improves the mAP by 0.4% and 0.7% for untrimmed videos of length $t = 5$ and $t = 10$ seconds, respectively, when compared to standard MIML on SF-50.

Impact of Actor-Action Assignment on untrimmed videos

We now analyze the impact of actor-action assignment on overall performance in Table 6.5. The proposed actor-action assignment results in 22.2%, 20.2% and 19.0% mAP for $t = 1$, $t = 5$ and $t = 10$ respectively with backbone of SF-50. This is an improvement of +1.4%, +1.8% and +1.9% mAP in comparison to standard miml

Backbone	video length	w/o assignment	Iteration 1	Iteration 2
SF-50	$t = 1$	21.8	**22.2**	22.1
	$t = 5$	18.8	19.8	**20.2**
	$t = 10$	17.8	18.8	**19.0**
SF-101	$t = 1$	24.0	**24.8**	23.8
	$t = 5$	21.1	21.8	**22.1**
	$t = 10$	19.0	19.4	**20.2**

Table 6.6: Improvement of results over number of iterations. The experiments are performed using all the scores during linear programming.

loss. It is well established that SF-101 is better than SF-50 for action recognition and video understanding (Feichtenhofer et al., 2019). With SF-101 backbone, our approach achieves state-of-the-art mAP of 24.8%, 22.1% and 20.2% for $t = 1$, $t = 5$ and $t = 10$ respectively, re-imphazing the need for powerful backbone even for weakly supervised action detection.

Performance variation across iterations

The proposed method for multi-label action detection iteratively alternates between model learning and action assignment, as shown in Figure 6.3. In Table 6.6, we evaluate the performance of this approach over multiple iterations with different backbones. For short videos of length $t = 1$, assignment initially improves the performance by +0.4% and +0.8% for SF-50 and SF-101, respectively. However, the performance decreases in the subsequent iteration, suggesting that explicit annotations for each person may be needed to improve results. This performance drop is more pronounced with the SF-101, likely due to its size (i.e. more number of parameters), which can amplify the negative effect of wrong assignments more quickly than SF-50. In contrast, for untrimmed video clips of length $t = 5$ and $t = 10$, the final performance is improved by +1.0% and +1.2% for SF-50 and SF-101 respectively, over multiple iterations. This indicates that multiple iteration of detection and assignment are needed to improve the performance for longer untrimmed videos.

Analysis of assorted subset scores

The actor-action assignment with $\varnothing$ uses assorted scores (Section 6.4.2) in the linear programming to achieve the state-of-the-art result. To quantify the effect of each individual scores in the overall method, we evaluate the method first with only a subset score ($p_{\omega,i,t}$). We then incorporate the temporal consistency score ($\tau_{\omega,i,t}$) and followed by self re-scaled score ($r_{\omega,i,t}$). All the experiments are done with SF-50 at $t = 1$, $t = 5$ and $t = 10$ setting. Table 6.7 tabulates the results with each of the components. $p_{\omega,i,t}$ achieves a mAP of 22.0%, 19.3% and 18.1%, respectively for

$p_{\omega,i,t}$	$\tau_{\omega,i,t}$	$r_{\omega,i,t}$	t = 1	t = 5	t = 10
Y	-	-	22.0	19.3	18.1
Y	**Y**	-	22.2	19.6	18.6
Y	**Y**	**Y**	**22.2**	**19.8**	**18.8**

Table 6.7: Result of the proposed approach with various parts of assorted subset scores as discussed in (6.11). The results are with SF-50 after only a single iteration

different values of t. $\tau_{\omega,i,t}$ is used to incorporate temporal consistency. The absolute gain in performance by +0.2%, +0.3% and +0.5% respective time t, showcases the impact of incorporating the temporal consistency. $r_{\omega,i,t}$ is used to re-scale the score of low scoring samples. Adding this only improves the performance by a minor margin of +0.2% for both $t = 5$ and $t = 10$. At the same time, $r_{\omega,i,t}$ doesn't improve the performance for $t = 1$. Based on the experiments, it is clear that temporal consistency is critical for better the actor-action assignment and subsequent improvement in action detection.

Impact of null-set assignment ($\varnothing$)

False positive person removals: The off-the-shelf person detector (Xie et al., 2017) can result in false detections. Our actor-action assignment helps in handling such false detections by assigning them with null-set ($\varnothing$), i.e. subset with action no-labels. This approach enables us to filter out these detections with null-set ($\varnothing$), ensuring better samples for subsequent training iterations. To evaluate our null-set assignment, we compare the mAP of the off-the-shelf person detections (only detection more than 0.8) to the mAP of all detections associated with one or more action labels across iterations, excluding those assigned to $\varnothing$ from the comparison. Our experiments, in the Table 6.8, show that our assignment approach increases mAP from 88.1% to 90.1% for short videos, while for longer untrimmed videos, the performance improves from 88.1% to 88.8% after multiple iterations. This increase in mAP highlights the effectiveness of the null-set assignment ($\varnothing$) in pruning false actor proposals. Without this null-set assignment and the resulting false-positive pruning, the person detection performance would remain constant at 88.1%.

Action Detection: Apart from the impact on pruning of some false positives in subsequent training, null-set ($\varnothing$) assignment also improves the overall performance in action detection. This is shown by the improvement of +0.4% mAP (in Table 6.8) after single iteration when compared to without null-set ($\varnothing$) assignment.

6.6.4 Comparison to state-of-the-art weakly supervised methods

Table 6.10 shows the comparison of the proposed approach with other weakly supervised methods. Our approach achieves state of the art results for both trimmed and untrimmed videos of various lengths as highlighted. $\mathcal{L}_{aaa+}$ achieves 22.1% and

method	length of video (t in secs)	Initial (Threshold of 0.8)	Iter 1	Iter 2
$\mathcal{L}_{aaa+}$ w/o $\varnothing$	1	88.1	88.1	-
$\mathcal{L}_{aaa+}$	1	88.1	90.1	-
$\mathcal{L}_{aaa+}$ w/o $\varnothing$	5	88.1	88.1	88.1
$\mathcal{L}_{aaa+}$	5	88.1	88.6	**88.8**

Table 6.8: Evaluation of False Positive Pruning: This table compares person-detections post label assignment showcasing the ability of the algorithm to prune false positive proposals. Experiments are conducted with SF-50 backbone on the training set.

Methods	
$\mathcal{L}_{aaa+}$ w/o $\varnothing$	19.4
$\mathcal{L}_{aaa+}$	**19.8**

Table 6.9: Results of actor-action assignment with and without null-set ($\varnothing$) on overall performance (results are presented after only 1 iteration of assignment). All the experiments are performed with SF-50 at $t = 5$

20.2%, beating other weakly supervised method (Arnab et al., 2020) with a gap of +4.1% and +4.4% for $t = 5$ and $t = 10$ respectively with untrimmed setting. As different backbones can have different impacts, we thus compare the results with same backbone. $\mathcal{L}_{aaa+}$ still achieves +2.2% and +3.2% mAP with same SF-50 for $t = 5$ and 10, respectively. This showcases the impact of iterative approach of assignment and retraining. For trimmed video (length $t = 1$), the $\mathcal{L}_{aaa}$ and $\mathcal{L}_{aaa+}$ achieves 23.1% and 24.8% mAP respectively with SF-101. The improvements in the performance of $\mathcal{L}_{aaa+}$, in comparison to $\mathcal{L}_{aaa}$, are due to the combined influence of negative training, null-set ($\varnothing$) assignment and temporal consistency. HGRNN (Chapter 5) explicitly incorporates the temporal context. We thus add HGRNN to the 3DCNN backbone. This further improves the performance by +0.8% for $\mathcal{L}_{aaa+}$ at trimmed video length of $t = 1$. These results indicate that the assignment approach is robust across various the backbone.

6.6.5 Comparison to fully supervised methods

In the end we compare the results of the proposed approach for $t = 1$ setting with fully supervised action detection in Table 6.11. Despite of having weak supervision, the proposed approaches are competitive to fully supervised approaches and even outperforms some of them (Sun et al., 2018, 2019; Girdhar et al., 2019). The proposed weakly supervised approach achieves 25.6% mAP, a relative performance of 77.3% to the state of the art fully supervised method that uses more powerful backbone and explicit action labels. Based on our experiments on various backbones,

Method	Backbone	$t = 1$	$t = 5$	$t = 10$
Uncertanity (Arnab et al., 2020)	SF-50	22.4	18.0	15.8
$\mathcal{L}_{aaa+}$	SF-50	22.1	20.2	19.0
$\mathcal{L}_{aaa}$	SF-101 + HGRNN	25.1	-	-
$\mathcal{L}_{aaa+}$	SF-101	24.8	**22.1**	**20.2**
$\mathcal{L}_{aaa+}$	SF-101 + HGRNN	**25.6**	-	-

Table 6.10: Comparison of the proposed method with state-of-the-art weakly supervised methods

we believe incorporating the recent state-of-art backbones to this approach, can further improve the performance of the proposed weakly supervised approach and thus reducing the gap to the state of the art fully supervised method.

6.7 Conclusion

In this chapter, we introduced the challenging task of weakly supervised multi-label spatio-temporal action detection with multiple actors. We first introduced a baseline based on multi-instance and multi-label learning. Furthermore, we presented a novel approach where the multi-label problem is represented by the power set of the action classes. In this context, we assign an element of the power set to each detected person using linear programming. Additionally, we introduced a temporal consistency measure within the constrained linear programming framework that favours consistent action labels across time. We evaluated our approach on the challenging AVA dataset where the proposed method outperforms the SOTA approach by a gain of $> 4\%$ mAP, especially over longer-length video clips. Despite of the weak supervision, the proposed approach is also competitive to fully supervised approaches.

Weakly Supervised Approaches		
Methods	Val	Test
Uncertainty (Arnab et al., 2020)	22.4	-
$\mathcal{L}_{aaa}$	25.1	23.5
$\mathcal{L}_{aaa+}$	**25.6**	-
Fully Supervised Approaches		
Methods	Val	Test
HGRNN (Chapter 5)	20.9	-
ATN (Girdhar et al., 2019)	25.0	24.9
LFB (Wu et al., 2019)	27.7	27.2
X3D (Feichtenhofer, 2020)	27.4	-
MViT (Fan et al., 2021)	28.7	-
Slowfast (Feichtenhofer et al., 2019)	29.4	-
HGRNN* (Chapter 5)	30.1	-
Slowfast++ (Feichtenhofer et al., 2019)	30.7	-
Object Transformer (Wu and Krahenbuhl, 2021)	31.0	-
AIA++ (Tang et al., 2020)	33.1	32.3

Table 6.11: Comparison to fully supervised approaches for $t = 1$ second settings. We also report the result of using HGRNN with full supervision and Slowfast-101. Note,"++" denotes the usage of multi-scale and horizontal flipping augmentation. Note HGRNN* is the HGRNN implementation with Slowfast-101.

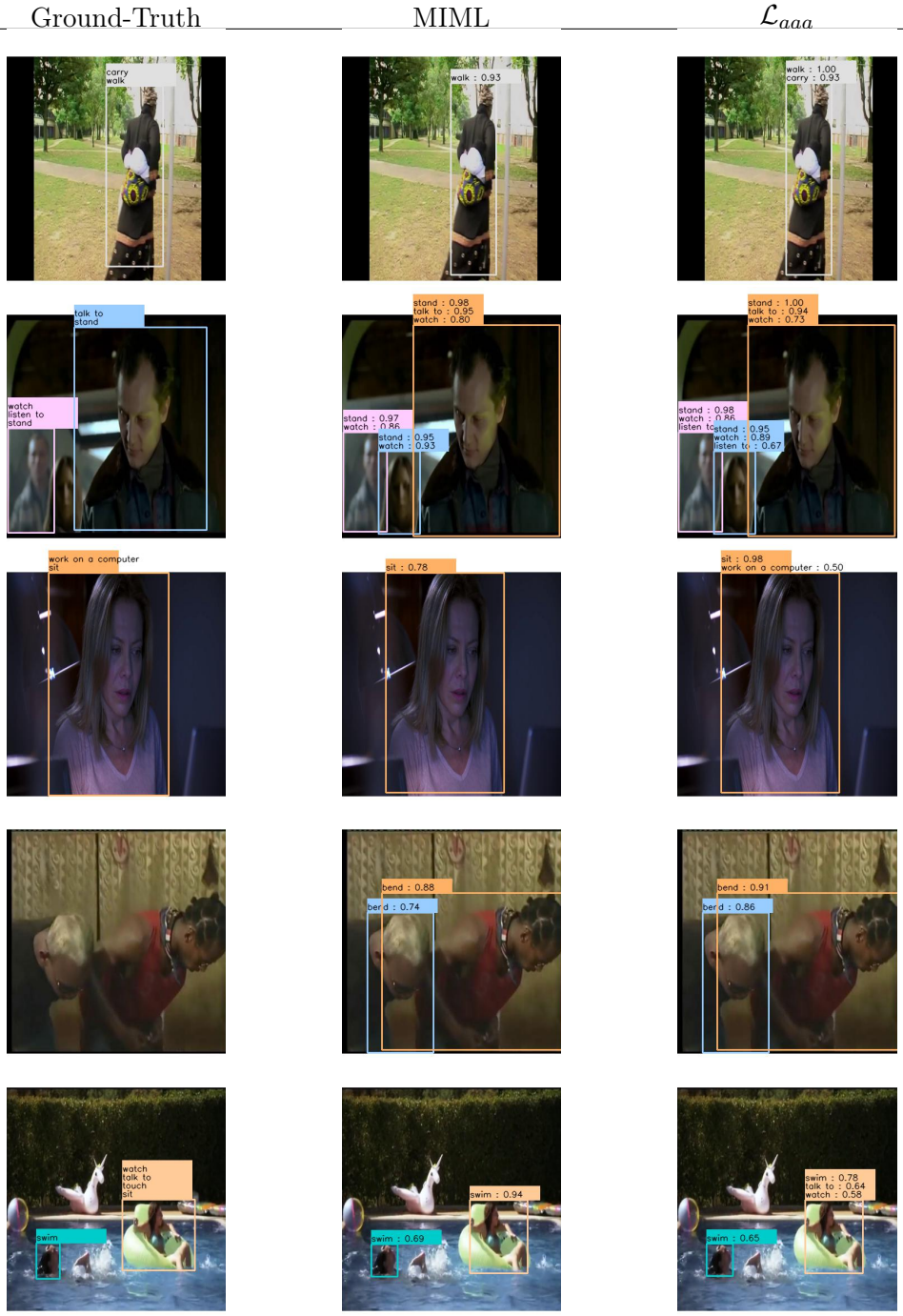


Figure 6.8: Qualitative results. The left column shows the ground-truth annotations. The middle column shows the results of the MIML baseline. The right column shows the results of the proposed method ($\mathcal{L}_{aaa}$). The colors distinguish only different persons, but they are otherwise irrelevant. The predicted action classes with confidence scores are on top of the estimated bounding boxes. The proposed approach recognizes more action classes per bounding box correctly compared to MIML. Both methods also detect genuine actions that are not annotated in the dataset as seen from the missing persons in the second and the fourth row. The bias of the proposed method towards the background is visible in last row, where the "swim" action is associated to both persons. Best viewed using the zoom function of the PDF viewer.

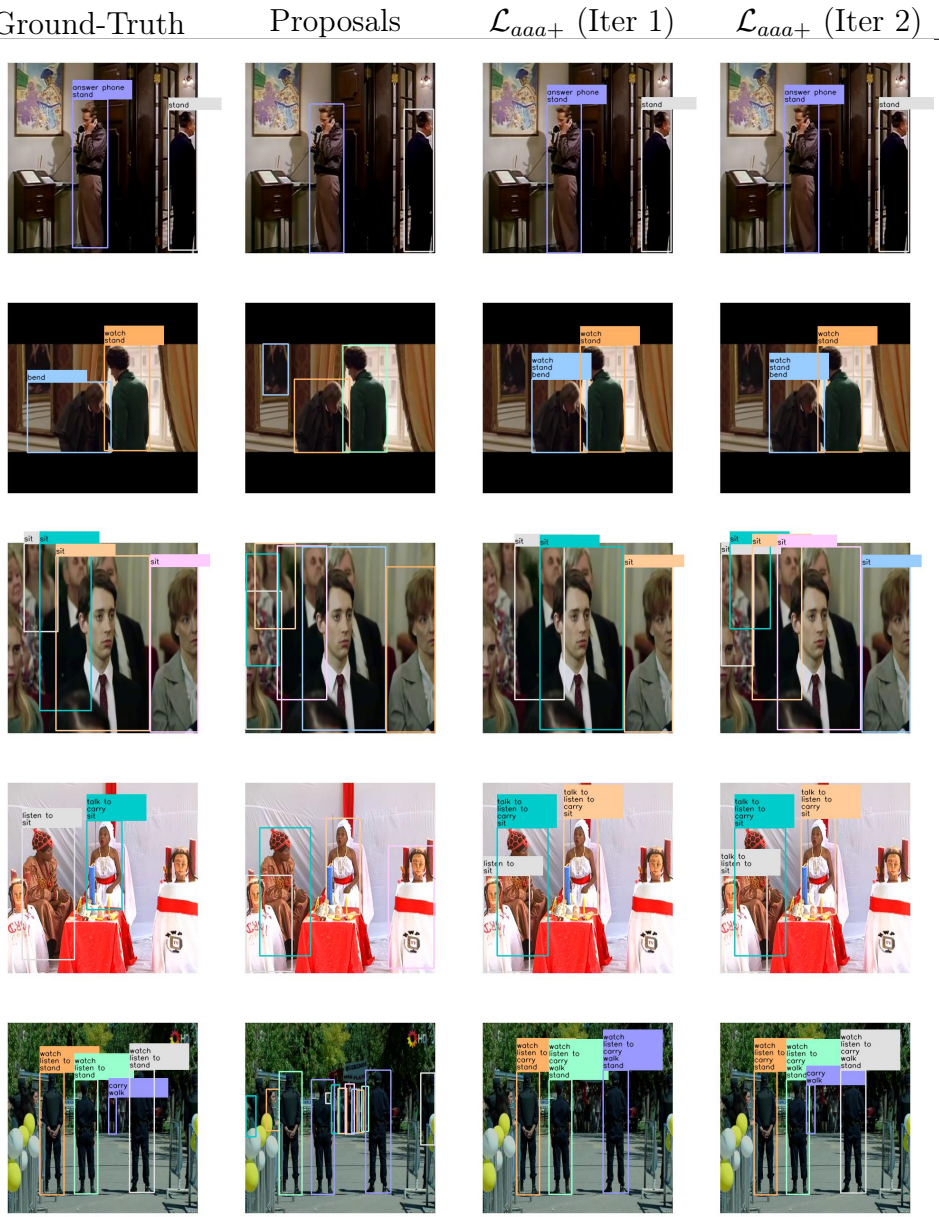


Figure 6.9: Qualitative results of label assignment and false-positive pruning. The leftmost column shows the ground-truth annotations of the training-set. The 2^{nd} column shows proposals used during the training. The 3^{rd} and 4^{th} columns showcase the labels after the assignment. Proposals with empty-set assignments are not displayed. The colors of the bounding boxes do not convey any information but are used to distinguish different persons. Best viewed using the zoom function of the PDF viewer.

Chapter 7

Multiple Instance Triplet Loss for Weakly Supervised Multi-Label Action Localisation of Interacting Persons

In Chapter 6, we introduced an iterative method for weakly supervised action detection that alternates between model training and actor-action assignment. Due to this iterative nature, the model training is slower and time-consuming. Furthermore, the label assignment without the null-set limits the use of the earlier approach only for trimmed videos. To tackle both these limitations, we introduce another method for weakly supervised action detection that uses multiple instance triplet loss for training in this chapter.

Individual Contribution

The following chapter is based on the publication (Biswas and Gall, 2021):

Multiple Instance Triplet Loss for Weakly Supervised Multi-Label Action Localisation of Interacting Persons
Sovan Biswas, and Jürgen Gall.
IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), 2021

This publication was done by Sovan Biswas and Jürgen Gall provided scientific guidance and supported this work with very valuable feedbacks and suggestions.

Contents

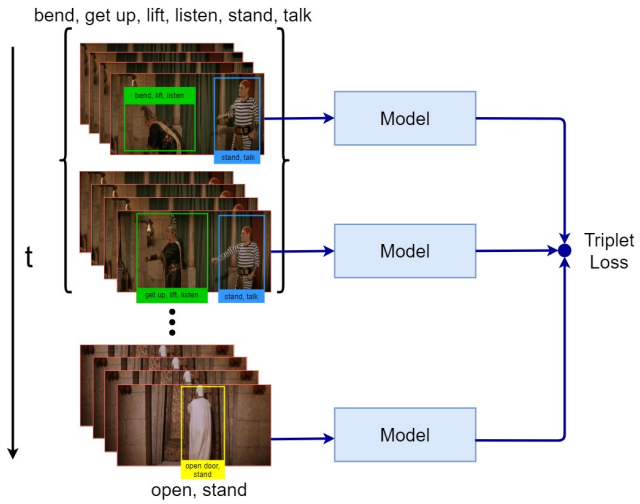


Figure 7.1: Overview of the Multiple Instance Triplet Loss (MITL) during training. A video shows a scene where two persons are interacting with each other. **Person A** indicated by the green bounding box performs the actions *bend*, *get up*, *lift* and *listen*, where the action *bend* changes to *get up*. **Person B** indicated by the blue bounding box performs the actions *stand* and *talk* consistently over time. These two frames build a positive pair of bags since there is at least one person (**person B**) that performs the same set of actions. For the negative bag, we sample more distant frames where there is at least one person that does not share any action with one person from the first frame. In this example, the **person C** indicated by the yellow bounding box performs the actions *open door* and *stand*, which are not performed by **person B**. Note that the bounding boxes are not provided during training and are only used for visualisation.

7.1 Introduction

Humans are multi-tasking by nature, i.e., they perform multiple actions such as reading, sitting, etc. simultaneously. Furthermore, they interact with each other in small groups. For instance, when a person talks, the other persons in the group listen. This led to a recent focus on multi-label action recognition datasets like (Gu et al.,

2018; Diba et al., 2020; Ji et al., 2020; Sigurdsson et al., 2016) and networks (Feichtenhofer et al., 2019; Girdhar et al., 2019) that can be applied to multi-label action detection problems.

One of the main limitations of these existing multi-label action detection approaches lies in the requirement for a high amount of annotated action labels and bounding boxes for action localisation. This requirement of high initial cost and human effort for building suitable models and algorithms limits the usage for large-scale real-world deployment. To circumvent the cost and time associated with data annotation for supervised training, new approaches for weakly supervised multi-label action detection in videos (Arnab et al., 2020) have been proposed. The focus of these approaches is on learning suitable models only from a set of actions that occur in a given video clip as seen in Figure 7.1 without any bounding box annotations. These weak annotations are much easier to obtain and reduce the cost and time associated with data annotation drastically. Even though an off-the-shelf person detector can provide very accurate location information, substantial challenges still exist to learn representations for action detection due to the lack of location-action association within the video clip. While this is not an issue for videos that show only one person as in (Jhuang et al., 2013), it is very challenging for videos where persons interact in small groups and perform multiple actions at the same time as in (Gu et al., 2018) and shown in Figure 7.1.

Weakly supervised multi-label action detection becomes even more challenging as the length of the video clips increases since the action labels are not provided per keyframe but per video clip. While in the initial part of Chapter 6 considered short video clips of 1 second, (Gu et al., 2018) proposed a protocol also for longer video clips. In this setting, the chances that an actor leaves or enters the scene, or changes the actions increase over time as shown in Figure 7.1. The approach (Arnab et al., 2020) deals with long video clips by subdividing a long video clip into multiple shorter clips. These shorter clips have the same action set annotation as the long video clip during training. This approach, however, discards temporal information like transitions between actions and temporal relations of actions. The accuracy thus decreases rapidly as the clips get longer.

In this chapter, we propose a novel approach for weakly supervised multi-label action detection that addresses these limitations for long video clips. The approach is inspired by the triplet loss that aims to learn a representation such that the similarity of a positive instance to an anchor instance is higher than the similarity of a negative instance to the anchor. In the weakly supervised and multi-label setting, however, it is not straightforward to build triplets consisting of an anchor, positive and negative instance as we do not know the corresponding actions of each detected person. We, therefore, extend the triplet loss to bags where a triplet consists of an anchor bag, a positive and a negative bag, and each bag contains all detected instances of a keyframe. The positive bag is from a keyframe that is temporally

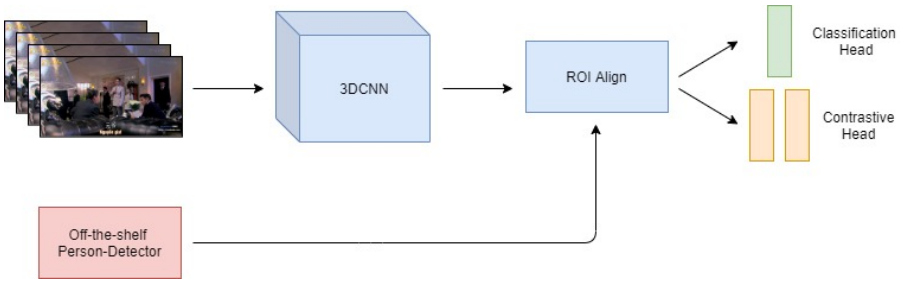


Figure 7.2: Overview of the network. We use a 3D CNN as backbone and a person detector to get bounding boxes. For each detected bounding box, we extract features from the 3D CNN using region-of-interest pooling. The features are passed through the *classification head* that predicts the class probabilities for the multi-instance and multi-label (MIML) loss and a *contrastive head* that predicts an embedding for the multi-instance triplet (MITL) loss.

close to the anchor keyframe whereas the negative bag is from a keyframe that is substantially distant as shown in Figure 7.1. For the multiple instance triplet loss, we define the similarities for positive and negative pairs of bags such that the loss selects the instances that are most consistent or least consistent, respectively. For instance in Figure 7.1, only person B (blue) of the positive pair of bags performs the same combination of actions in both keyframes while one action of person A (green) changes. For the negative pair, only person C (yellow) and person A do not share any common action while person C and person B both stand. We evaluate the proposed approach on the challenging AVA 2.2 dataset (Gu et al., 2018), where we outperform the state-of-the-art for weakly supervised multi-label action localisation for training on video clips between 5 and 30 seconds.

7.2 Weakly Supervised Multi-Label Action Localisation

The task of multi-label action localisation requires to detect and recognise all actions that are performed by each person in a video as shown in Figure 7.1. In contrast to standard action localisation, where it is assumed that one person performs only one action in a video clip, here each person can perform multiple actions at the same time. Further, the actions, as well as the number of persons that are performing these actions, can vary over time as shown in Figure 7.1. Nevertheless, we can assume some temporal consistency where at least a subset of the actions is continued in neighbouring frames. Recently, approaches for multi-label action localisation have been proposed that can be trained with weak supervision (Arnab et al., 2020), i.e., without any bounding box annotations for the training videos. In this setting, only the set of actions performed by all persons occurring in a video is provided as

annotation. In this work, we follow this protocol for weakly supervised multi-label action localisation, but we focus on learning from temporal consistency.

To exploit the temporal consistency, we propose a triplet loss across time for training. It requires to define triplets (s, p, n) where the positive sample p contains the same actions across time as s and the negative example n contains different actions. Despite the temporal consistency assumption, we cannot assume that all actions are consistent as shown Figure 7.1. We also do not know who is performing the given actions in the training video due to the weak supervision. We therefore propose a multiple instance triplet loss that is computed for bags of instances (S, P, N) as shown in Figure 7.3, instead of single instances (s, p, n). The loss then aims to minimise the distance between S and P, which will be defined for bags instead of instances, and maximise the distance between S and N. We will first describe the network architecture, which is illustrated in Figure 7.2, in Section 7.2.1 and then describe the novel multiple instance triplet loss, which is illustrated in Figure 7.3, in Section 7.2.2. Finally, we summarise the entire loss function in Section 7.2.3.

7.2.1 Network

As the objective is to detect actions, we first generate multiple proposals to locate various actors similar to (Arnab et al., 2020) and Chapter 6. The proposals are generated using an image-based off-the-shelf person detector based on Faster-RCNN (Ren et al., 2015) using the ResNeXt backbone (Xie et al., 2017). We denote the detected person proposals at time t by $A_t = \{a_i\}$ where a_i is the i^{th} person and A_t is the set of all detected persons at t. As in (Gu et al., 2018; Girdhar et al., 2019), we then use these person locations to generate person-specific representations from a 3D CNN by applying 2D region of interest pooling at the same spatial location for a temporal window of 1 second as shown in Figure 7.2.

These person specific features are passed through two different heads, namely a *classification head* $f(\cdot)$ and a *contrastive head* $g(\cdot)$, as shown in Figure 7.2. The classification head consist of a single layer MLP with a sigmoid activation function σ to predict the probabilities for each class, i.e., $f(a_i) = \sigma(W_{class} x_i)$, where x_i is the person specific feature representation from the 3D CNN and $W_{class} \in \mathbb{R}^{D \times N}$ are the weights of the layer. D is the dimension of the feature representation and N is the number of action classes. The contrastive head consists of an MLP with two layers similar to (Chen et al., 2020), i.e., $g(a_i) = \text{norm}(W_2 \, \text{ReLU}(W_1 x_i))$, where $W_1 \in \mathbb{R}^{D \times 512}$, $W_2 \in \mathbb{R}^{512 \times 512}$, and norm denotes L2 normalisation.

7.2.2 Multiple Instance Triplet Loss

The objective of contrastive learning is to learn a representation such that similar instances are close to each other, while dissimilar ones are far apart. One common

loss for contrastive learning is the triplet loss:

$$\mathcal{L}_{triplet}(s, p, n) = \max(0, \text{sim}(s, n) - \text{sim}(s, p) + \alpha) \tag{7.1}$$

where $\text{sim}(i, j) = g(a_i)^T g(a_j)$ is the cosine similarity of the L2 normalised embedding for the detected persons a_i and a_j with $-1 \leq \text{sim}(i, j) \leq 1$. α denotes the margin between the positive pair (s, p) and the negative pair (s, n).

In the context of weakly supervised learning, we do not know the labels of the detected bounding boxes a_i and it is therefore not straightforward to generate triplets of instances where a_p contains the same actions as a_s whereas a_n contains different actions. We therefore propose to extend the triplet loss (7.1) to bags, which contain multiple instances. While we describe in Section 7.2.2.1 how an anchor bag S and a corresponding positive bag P and negative bag N are selected, we define the multiple instance triplet loss for a triplet of bags (S, P, N) by

$$\mathcal{L}_{triplet}(S, P, N) = \max(0, \text{sim}_\text{n}(S, N) - \text{sim}_\text{p}(S, P) + \alpha) \tag{7.2}$$

where $\text{sim}_\text{n}(S, N)$ defines the similarity for a negative pair of bags and $\text{sim}_\text{p}(S, P)$ the similarity for a positive pair. We will describe the similarity measures for bags in Section 7.2.2.2.

7.2.2.1 Triplet Selection

In order to build triplets of bags (S_t, P_t, N_t) for each frame t in the training videos, we first take all detections of the frame t as anchors, i.e., $S_t = A_t$. Since persons are likely to perform similar actions over a short period of time as shown in Figure 7.1, we select randomly frame $t_p \in \{t-1, t, t+1\}$ and take all detections in frame t_p as positive bag, i.e., $P_t = A_{t_p}$. In case of $t_p = t$, a random transformation like random cropping and mirroring is applied such that P_t is not exactly the same as S_t. For the negative bag N_t, a random frame t_n with $|t_n - t| \geq 100$ is selected and $N_t = A_{t_n}$. Since there is a large temporal gap between N_t and S_t, it is very unlikely that all persons perform the same actions in frame t and t_n.

However, we cannot assume that all persons in S_t and P_t perform the same set of actions over time as shown in Figure 7.1. Similarly, it might be possible that persons in S_t and N_t perform some common actions. While it is in principle possible to select negatives from other videos that do not share any actions, this does not provide many negatives for the AVA 2.2 dataset (Gu et al., 2018) since basic actions like *stand* occur in most frames and videos. We therefore need to define the similarity measures $\text{sim}_\text{n}(S, N)$ and $\text{sim}_\text{p}(S, P)$ that are robust to missing and overlapping actions, which we will discuss next. For the ease of reading, we omit the frame index t and denote triplets by (S, P, N).

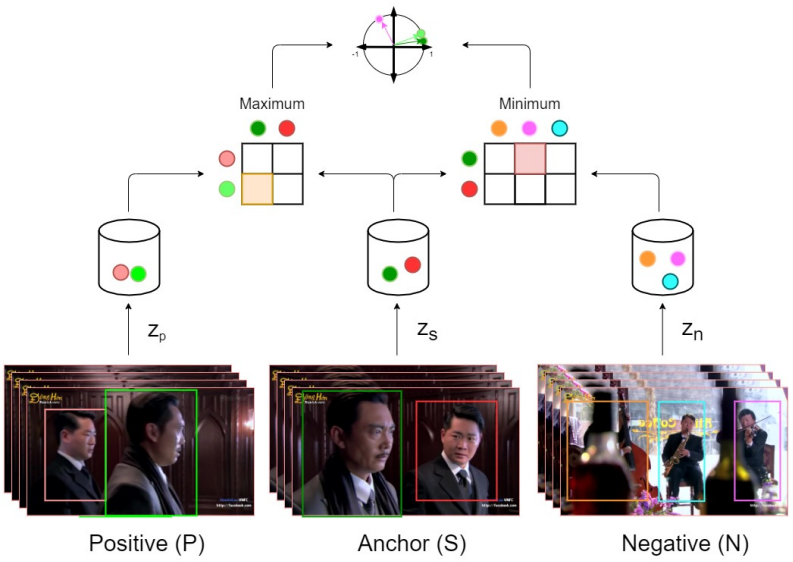


Figure 7.3: Illustration of the Multiple Instance Triplet Loss (MITL). Any triplet (S, P, N) comprises an anchor S, positive P and negative N bag, where each bag contains a set of detected bounding boxes. As the anchor and positive bag are from the temporal neighbourhood, we assume that they contain at least one instance that is consistent in both bags. This is found by the maximum similarity among the persons in the two bags. The anchor and negative bag are temporally distant. Nevertheless, they can share persons performing the same actions. We thus assume that there is at least one pair of instances that are dissimilar, which is obtained by the minimum similarity. The Multiple Instance Triplet Loss (MITL) thus aims to minimise the distance between the green instances from S and P and to maximise the distance between the green instance from S and the pink instance from N.

7.2.2.2 Similarity of Bags

While we cannot assume that each person in S performs the same set of actions in P or is even present in P, we assume that there is at least one close match such that $a_s \in S$ and $a_p \in P$ should be similar as shown in Figure 7.1 and Figure 7.3. For instance, the person in the blue bounding box in Figure 7.1 continues the same actions, while the person in the green bounding box changes one of the three actions. In order to measure the similarity between the bags S and P, we therefore only consider the best match. In addition, we also consider that persons performing the same actions are spatially consistent. We thus define sim_p by

$$\text{sim}_\text{p}(S, P) = \max_{a_s \in S, a_p \in P} \left\{ \exp(-\|l_s - l_p\|_2^2) g(a_s)^T g(a_p) \right\} \tag{7.3}$$

where l_s and l_p are the centroid locations of the detections a_s and a_p, respectively, and $g(a_s)^T g(a_p)$ is the cosine similarity of the L2 normalised embedding for a_s and a_p.

For the negative pair of bags (S, N), we have to consider the possibility that persons $a_s \in S$ and $a_n \in N$ perform the same actions. We therefore assume that there is at least one pair of persons in S and N that do not perform the same combination of actions. We thus define sim_n by

$$\text{sim}_\text{n}(S, N) = \min_{a_s \in S, a_n \in N} g(a_s)^T g(a_n). \tag{7.4}$$

7.2.3 Loss Function

In order to train the network, we use besides the multiple instance triplet loss $\mathcal{L}_{triplet}$ two additional loss functions:

$$\mathcal{L} = \mathcal{L}_{miml} + \mathcal{L}_{triplet} + \mathcal{L}_{sim} \tag{7.5}$$

where $\mathcal{L}_{MIML}$ is a loss for multi-instance and multi-label learning (7.6) and $\mathcal{L}_{sim}$ is a similarity loss (7.7) that encourages that the absolute value of two similar samples is high. We add the additional loss functions without weighting them.

The multiple instance triplet loss does not take any supervision into account. Instead we build triplets based on temporal proximity. In order to train the network using the set of labels that is provided for each video clip (Arnab et al., 2020) and contains all actions that are performed by all persons in a video clip, we use a loss for multi-instance and multi-label (MIML) learning (Nguyen et al., 2013; Nguyen, 2010; Yang et al., 2017). The MIML loss 6 is defined by

$$\mathcal{L}_{miml} = \mathcal{L}\left(Y, \max_i(f(a_i))\right) \tag{7.6}$$

where Y is a vector that is one for all classes that are present in the video clip and zero otherwise, $f(a_i)$ is the vector that contains the predicted class probabilities for the detection a_i, and max is the class-wise maximum over all detections. $\mathcal{L}$ is the binary cross entropy.

While the triplet loss encourages the network to learn a representation in order distinguish persons performing similar or dissimilar actions and the margin α can be increased to increase the difference, we show in the experiments that the accuracy decreases when α is too large. For small values of α, however, the absolute similarity for correct pairs can remain low as long as the difference between positive and negative pairs is larger than the margin α. In order to also encourage that the absolute similarity of positive pairs is larger equal to β, we add the similarity loss:

$$\mathcal{L}_{sim} = \max(0, \beta - \text{sim}_\text{p}(S, P)). \tag{7.7}$$

In the experiments, we evaluate the impact of the loss terms.

7.3 Experiments

7.3.1 Dataset

We evaluate our approach on the AVA 2.2 dataset (Gu et al., 2018). The dataset contains 235 videos for training and 64 videos for evaluation. Each video is 15 minutes long. The annotation is provided for each person with 60 action classes and bounding box locations at keyframes with a sample rate of 1Hz for the fully supervised setting. The accuracy is measured by mean average precision (mAP) over all actions with an IoU threshold of 0.5 at the key-frames as described in (Gu et al., 2018). In the weakly supervised setting, the bounding box information is not used during training. Following the protocol of (Arnab et al., 2020), the training videos are subdivided into t = 1, 5, 10, and 30 keyframes, i.e., clips of 1, 5, 10, and 30 seconds, respectively. For each clip, the list of actions that are present in the keyframes is provided. This means that the protocol for $t = 30$ is much more difficult than the protocol for $t = 1$ since we do not know when and where the actions occur in the video clip.

We use Faster RCNN (Ren et al., 2015) with ResNeXt-101 (Xie et al., 2017) as backbone to detect persons. This detector is pre-trained on ImageNet and fine-tuned on the COCO dataset. We perform our experiments with SlowFast-50 and SlowFast-101 pre-trained on Kinetics 600. The temporal scope for SlowFast (SF) was set to 64 frames with a stride of 2. We used random cropping and flipping for data augmentation. We randomly crop images of size 224×224 pixels from the resized frame of 256 pixels at its shorter side.

7.3.2 Comparison to the state-of-the-art

Table 7.1 shows the comparison of the proposed approach with other state-of-the-art methods. The proposed approach outperforms the state-of-the-art method (Arnab et al., 2020) by +2.7%, +4.0% and +5.8% for untrimmed videos of length $t = 5$, 10 and 30 seconds, respectively. This gain can be partly attributed to the better 3D CNN. Thus, we compare the approach with the same Slowfast-50 as (Arnab et al., 2020). With the same backbone, the proposed approach is still +1.3%, +1.8% and +4.2% better for untrimmed videos of length t = 5, 10 and 30 seconds, respectively. This gain shows that the proposed approach resolves temporal ambiguity in untrimmed videos. The approach performs better as the length of the untrimmed video increases as seen by the increase in mAP difference compared to (Arnab et al., 2020). Interestingly, the proposed approach achieves a lower accuracy for trimmed

Code: `https://github.com/sovan-biswas/MITL`

Methods	3D-CNN	$t = 1$	$t = 5$	$t = 10$	$t = 30$
Uncertainty (Arnab et al., 2020)	SF-50	22.4	18.0	15.8	11.4
Actor-Action (Chapter 6)	SF-50	21.6	-	-	-
Actor-Action (Chapter 6)	SF-101	**25.1**	-	-	-
Proposed Approach	SF-50	20.7	19.3	17.6	15.6
Proposed Approach	SF-101	22.1	**20.7**	**19.8**	**17.2**

Table 7.1: Comparison of the proposed method with other state-of-the-art methods. The clip length of the training videos with weak annotations is denoted by $t = 1$, 5, 10, and 30 seconds. The larger t is, the more difficult is the task.

$\mathcal{L}_{miml}$	$\mathcal{L}_{triplet}$	$\mathcal{L}_{sim}$	$t = 5$	$t = 30$
Y	-	-	18.6	14.4
Y	**Y**	-	-	14.9
Y	**Y**	**Y**	**19.3**	**15.6**

Table 7.2: Impact of loss functions for $t = 5$ and 30 seconds. $\mathcal{L}_{miml}$ is the MIML loss, $\mathcal{L}_{triplet}$ is the multiple instance triplet loss and $\mathcal{L}_{sim}$ is the similarity loss. **Y** denotes that the loss function has been used.

videos of $t = 1$ seconds. With Slowfast-50, it is by -1.7% and -0.9% mAP lower compared to (Arnab et al., 2020) and Chapter 6, respectively. For $t = 1$ second, the actions for each keyframe are provided such that a temporal association is not required and the provided labels only need to be assigned to the persons in each keyframe. This is explicitly addressed in Chapter 6 and the method thus achieves the highest accuracy for $t = 1$ i.e. trimmed videos. The approach, however, cannot be directly applied to a setting with $t > 1$ and needs modification as discussed in Chapter 6 for untrimmed videos. In contrast, the multiple instance triplet loss and the similarity loss are only suitable for a setting with $t > 1$.

7.3.3 Ablation Studies

7.3.3.1 Impact of Loss Functions

In (7.5), we use the loss functions $\mathcal{L}_{miml}$, $\mathcal{L}_{triplet}$ and $\mathcal{L}_{sim}$. While $\mathcal{L}_{miml}$ is always required since it is the only loss function that takes the weak annotations into account, Table 7.2 shows the quantitative impact of the other loss functions for $t = 5$ and 30 seconds, respectively. $\mathcal{L}_{triplet}$ increases mAP by $+0.5\%$ at $t = 30$ seconds, indicating the impact of contrastive learning in removing temporal ambiguity in long untrimmed videos. When both $\mathcal{L}_{triplet}$ and $\mathcal{L}_{sim}$ are used, mAP increases by $+0.7\%$ and $+1.2\%$ mAP for $t = 5$ and 30 seconds, respectively. $\mathcal{L}_{sim}$ ensures that the absolute similarity of positive matches over time is large.

α	0	0.05	0.1	0.3
Overall	15.1	**15.6**	15.6	14.5
Least 10	1.7	1.9	2.7	**2.8**
Top 5	55.4	**55.4**	55.3	54.2

Table 7.3: Impact of the margin α for $t = 30$ seconds. The Least 10 and Top 5 indicate the least 10 frequently and top 5 frequently occurring classes in the training dataset.

β	1	0.8
Overall	13.9	**15.6**

Table 7.4: Impact of the margin β for $t = 30$ seconds.

7.3.3.2 Impact of α

In (7.2), α causes the model to separate negative and positive bags, and as α increases the distance becomes larger. Due to the multi-label scenario, persons often share some actions as shown in Figure 7.1. If a positive pair does not share all actions or a negative pair shares some actions, a large value of α can have a negative impact. In Table 7.3, we thus quantitatively analyse the impact of α using SlowFast-50 as backbone for the $t = 30$ setting. As expected, the accuracy decreases for $\alpha = 0.3$ and the best mAP of 15.6% is obtained for α between 0.05 and 0.1. In all other experiments, we use $\alpha = 0.05$. Interestingly, the approach is able to classify the least 10 frequently occurring classes better with $\alpha = 0.3$ as seen by an increase of mAP to 2.8%. This is in contrast to the deteriorating performance of the top 5 frequently occurring classes with an increasing value of α. This is due to the fact that frequently occurring classes are more likely to be present both in an anchor and negative bag and are thus more likely to be shared by a negative pair.

7.3.3.3 Impact of β

The loss (7.7) encourages that the absolute similarity of positive pairs is larger equal to β. While we evaluated the impact of the loss already in Table 7.2, we evaluate the impact of β in Table 7.4 for $t = 30$ seconds. In case of $\beta = 1$, the loss aims to maximise the absolute similarity since sim_p cannot be larger than 1. This reduces the accuracy compared $\beta = 0.8$, which we use in our experiments.

7.3.3.4 Variants of sim_n

As we discussed in Section 7.2.2.2, we assume that there is at least one pair of persons in the anchor and negative bag that do not perform the same combination of actions and thus take the minimum over all pairs to compute sim_n (7.4). If we take the maximum instead of the minimum, we would assume that none of the pairs

Negative-Bag Similarity	max	min
Overall	14.9	**15.6**
Least 10	1.7	**1.9**
Top 5	54.7	**55.4**

Table 7.5: Comparison of taking the min or max to compute sim_n (7.4) for $t = 30$ seconds. Least 10 and Top 5 indicate the least 10 frequently and top 5 frequently occurring classes in the training dataset.

shares any action. In Table 7.5, we compare the maximum and the minimum using SlowFast-50 as backbone for the $t = 30$ setting. The approach achieves mAP of 15.6% for the min and 14.9% for the max operation. Taking the minimum for the negative pairs of bags also outperforms the maximum for the least 10 and top 5 occurring classes. However, the gain of min is larger for the top 5 occurring classes since they are more likely to be shared by a negative pair.

7.4 Conclusion

In this chapter, we proposed a novel approach based on contrastive learning for weakly supervised multi-label action localisation. In this setting, only the list of actions occurring in each training video is provided. So, in order to learn a better representation despite weak annotation, we introduced the novel Multiple Instance Triplet Loss (MITL) which takes the similarity of bags instead of instances into account. Later, we evaluated our proposed approach on the challenging AVA dataset, where it is difficult to define negative pairs since some actions like standing occur very frequently. We therefore addressed this issue by defining different similarity functions for positive and negatives bags. For the setting where the training videos are longer than 1 second, our approach achieved state-of-the-art results.

Chapter 8

Conclusion

Contents

In this chapter, we conclude the thesis by summarizing the contributions toward modeling human actions in multi-label settings. Ultimately, we discuss the possible future research directions and exploration needed to improve the state-of-the-art in respective tasks.

8.1 Summary and Contributions

In this thesis, we have delved into the intricacies of modeling human actions in multi-label settings, a challenging task in computer vision. With the ever-increasing prevalence of video data and the growing demand for real-time action recognition and localization, accurately identifying and interpreting multi-label actions are crucial for many applications.

We have presented novel approaches that use RNNs, CNNs, and GNNs to tackle this challenge effectively. These techniques have been extensively evaluated on benchmark datasets, outperforming existing methods in various scenarios.

The primary contributions of this thesis can be categorized into two sections as follows:

- Group Activity Analysis
- Multi-Label Action Detection and Recognition

8.1.1 Group Activity Analysis

As outlined in Section 1.2, human actions in group settings exhibit two distinct characteristics: individual actions and group interactions. To effectively capture

these characteristics, we proposed a modified Structural Recurrent Neural Network (SRNN) architecture (Chapter 4), which explicitly models individual actions and the interactions between individuals. The approach drastically differs from existing two-stage hierarchical recurrent neural networks (RNNs) for group activity analysis that suffers from a lack of joint group and individual action training and a lack of ability to capture interactions between individuals. This lack of joint analysis leads to suboptimal performance. The SRNN architecture is based on a graph representing individuals as nodes and interactions between them as edges. The SRNN architecture uses interconnected RNNs to process the information within the graph. In Chapter 4, we proposed two variations of SRNN architecture: SRNN-max node and SRNN-max edge. SRNN-max node focuses on pooling node characteristics, whereas SRNN-max pools edge characteristics for group activity recognition.

Our results on the Volleyball dataset demonstrate that the SRNN architecture outperforms previous methods in accuracy. In particular, the SRNN-max node variant, which focuses on pooling hierarchical node features to recognize the group action, achieves the best results. This is because the SRNN-max node variant can pool the relevant information about the key actor in a group activity. For example, in the case of a "left spike" in volleyball, the key individual action is "spiking" on the left side of the volleyball field.

8.1.2 Multi-label Action Detection and Recognition

Humans are naturally multi-taskers, capable of performing multiple actions simultaneously. Unlike recognizing actions in short clips with only one action label, we focus on recognizing multiple actions that co-occur within a video.

8.1.2.1 Modelling Scene and Spatial dependencies of actions

Actions have several types of dependencies: temporal, exclusionary, and interactive. As seen in group activity analysis, temporal dependencies capture the long-term scene relationships, while interactional dependencies capture the relationships between humans. Exclusionary information captures the improbability of specific actions, such as "driving" and "standing," occurring together.

In Chapter 5, our proposed approach is a hierarchical model that models the temporal scene using an RNN at the lower level and the occurrence of simultaneous actions, exclusions of actions, and interactions of detected persons at the top level using a graph-RNN. All detected persons in a frame represent the nodes of the graph-RNN. The lower-level scene RNN and higher-level graph RNN are trained jointly.

The proposed approach achieves high results on the AVA multi-label action dataset by capturing the temporal, exclusionary, and interactive context of actions. We also demonstrated that using longer temporal scene RNN improves scene con-

text and representation of both persons and frames and is essential to significantly improved performance.

8.1.2.2 Weakly Supervised Approaches for Multi-Label Action Localisation

Traditional action recognition approaches require extensive person-specific action class *annotations* for training. These annotations are time-consuming and expensive to obtain. This limitation hinders the development of real-world applications that require quick deployment and adaptation. To address this challenge, we explored weakly supervised learning methods for multi-label action recognition (Chapter 6 and Chapter 7), where only a subset of video frames are annotated with action labels. The weakly supervised learning framework is based on the multi-instance and multi-label learning (MIML) paradigm. In MIML, each video clip containing multiple person instances is treated as a bag and a list of actions as the weak annotation. This framework allows efficient training of action recognition models without requiring detailed per-person annotations.

The first approach employs an iterative strategy alternating between training the multi-label action detection model and inferring actor-action associations. The actor-action association problem is formulated as an assignment task, where the goal is to assign the most likely actions to each detected person based on their features and the video's action annotations. This assignment is solved using linear programming. Our experiments show that multiple iterations of the training and inference process lead to significant performance improvements. This iterative approach effectively leverages the information extracted from the action detection model to refine the actor-action associations, ultimately improving the overall recognition accuracy.

The second approach further enhances the MIML framework by incorporating contrastive learning. A contrastive loss compares anchor, positive, and negative bags. *Positive bags* are defined as those with high temporal similarity to the anchor bag, while negative bags have low temporal similarity. This contrastive loss mechanism encourages the model to learn more robust representations of actions, even when dealing with weaker annotations over longer video durations.

Both proposed weakly supervised learning methods demonstrate competitive performance compared to fully supervised approaches, especially when annotating over shorter durations. It highlights the effectiveness of the weakly supervised learning framework in reducing the annotation burden while maintaining high recognition accuracy.

8.2 Outlook

Research on recognizing and detecting multi-label human activities is developing quite rapidly. While many of the proposed approaches in this area have shown good

performance, reliably deploying them in real-world critical applications requires more improvements and addressing their current limitations. We now briefly discuss some of the open challenges and possible extensions to the proposed approaches in this thesis.

8.2.1 Advancing Group Activity Analysis

8.2.1.1 Approaches based on Transformer and Attention

Our group and individual activity analysis approaches used Recurrent Neural Networks and Convolutional Neural Networks. These approaches are easy to implement and deploy and suitable for small-scale data. However, modern transformer architectures are more powerful and effective, resulting in increased performance. Transformers are effective due to their multi-headed attention. This could be key as not all pairs of interactions are equally significant in group activities. Thus, using attention can result in prioritizing the significant interactions for individual and group activity recognition. Similar ideas with attention have been explored in (Yuan et al., 2021; Li et al., 2021). Transformer-based architectures require a large amount of data to learn effectively. As such, variations of self-supervised contrastive learning have been used in (Chappa et al., 2023) to improve the performance.

8.2.1.2 Weakly supervised and unsupervised learning

One of the critical issues for the real-world deployment of algorithms is the availability of annotations. It is primarily an issue for group activities where annotations are required for every person within the group. As mentioned in Section 1.2, most people within a group perform the same individual action. Thus, approaches that require a sparse or weak list of annotations can be explored using the assumption that most people will have the same individual actions. Recently, (Zappardino et al., 2021) explored group activity analysis without individual labels. Due to the high likelihood of various people performing the same individual action in a group, unsupervised and self-supervised approaches can be explored in the future.

8.2.1.3 Group activity analysis based on textual descriptions

Modern models and approaches to person detectors are highly accurate. Furthermore, the recent exploration with weak supervision and multimodal supervision (Zappardino et al., 2021) has shown much promise. A critical future direction to explore is to perform group activity analysis from human descriptions. This is of high importance as the availability of large-scale data with detail annotation remains a significant bottleneck, as explained before. However, one can easily retrieve group videos with textual descriptions through movie dialogues or sports commentary. Developing group activity analysis with such noisy and sparse textual descriptions

would pave the way for real-world deployment of these algorithms in new and previously unexplored applications. Recently, (Mkhallati et al., 2023) generated textual descriptions based on group action analysis that can be used as a substitute for commentary.

8.2.2 Advancing Multi-label Activity Recognition and Detection

8.2.2.1 Short-term temporal action consistency

In Chapter 6, we explored the consistency of action labels over a short period. However, our approach and consistency exploration were limited to the label space without explicit matching of actors and the similarity of features. Thus, exploring models and loss functions that ensure temporal consistency in the feature space across two frames can improve the performance of multi-label action recognition and detection. One of the key issues in such problems is identifying the same actors across two different frames, as the actors could have moved locations across frames. Recognizing and tracking persons can help identify the same person in two nearby timestamps. The authors of (Vicol et al., 2018) and (Kukleva et al., 2020) explore this research direction.

8.2.2.2 Long-term Scene Relations

Videos are seldom short, as in typical clip-based action recognition (Carreira and Zisserman, 2017). Mostly, videos are long, and they have long-term temporal dependencies. It is apparent during movies an actor is present in different scenes with some temporal dependencies. These scenes can occur more than 10 minutes apart. Thus, modeling such long-term dependencies can improve multi-label action detection and recognition. In (Vicol et al., 2018), the authors show that such temporal relationships exist over a significantly longer time than the capability of current approaches.

8.2.2.3 Multi-Label Activation Functions

Current approaches (as also used in Chapter 5, Chapter 6 and Chapter 7) that perform multi-label action recognition typically utilize *Sigmoid* as activation function over the label space. Though simple, it performs poorly during class imbalance. Furthermore, applying label space constraints, such as 'sit' and 'stand', which can not co-occur on the same person, is challenging. It requires further exploration towards better activation functions that tackle the class imbalance and label space constraints. In (Kerrigan et al., 2021), the authors use the label space constraints in the zero-shot implementation.

8.2.2.4 Attention based architectures for label dependencies

In Chapter 5, we exploited the human and scene dependencies for multi-label action recognition. However, all the dependencies in our approach were weighted equally and pooled using a max pool operation. It is a clear limitation as each action label within the multiple labels can have different priority label dependencies. For example, a person who is *walking*, *carrying a bag*, and *listening* have different dependencies. *Walking* and *carry a bag* have temporal dependencies whereas *listening* have spatial dependency of another person speaking. Thus, label-specific attention can disambiguate the dependencies and improve representation for better performance. In (Kovtun et al., 2023), the authors explore the idea in label space for future customer purchase prediction.

8.2.2.5 Multi-modal Models

Actions such as *speaking* and *listening* have limited visual cues. As such, they are difficult to disambiguate, primarily with multi-label activation functions such as *sigmoid*. Incorporating modalities such as speaker and audio information can help disambiguate these classes. With this in mind, the AVA-active speak dataset (Zhang et al., 2019) was formed, showing improved performance than on the AVA dataset.

Recently, large multimodal models (Alayrac et al., 2022) are being developed as foundational models. These models can provide additional context for zero-shot learning or improve the performance of existing models when trained on limited annotations. It is very effective as specific actions, though not distinguishable in visual embedding space, are significantly different in textual embedding space. For example, in (Kerrigan et al., 2021), the authors use a textual modality for zero-shot learning.

Bibliography

Adler, I., Resende, M. G., Veiga, G., and Karmarkar, N. (1989). An implementation of karmarkar's algorithm for linear programming. *Mathematical programming*, 44:297–335. 26

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Li, F., and Savarese, S. (2016). Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971. 7, 30

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K. (2022). Flamingo: a visual language model for few-shot learning. *CoRR*, abs/2204.14198. 94

Amer, M. R., Lei, P., and Todorovic, S. (2014). Hirf: Hierarchical Random Field for Collective Activity Recognition in videos. In *European Conference on Computer Vision (ECCV)*, pages 572–585. 8

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., and Schmid, C. (2021). Vivit: A video vision transformer. In *International Conference on Computer Vision (ICCV)*, pages 6836–6846. 20

Arnab, A., Sun, C., Nagrani, A., and Schmid, C. (2020). Uncertainty-Aware Weakly Supervised Action Detection from Untrimmed Videos. In *European Conference on Computer Vision (ECCV)*, pages 751–768. 56, 64, 72, 73, 74, 79, 80, 81, 84, 85, 86

Bagautdinov, T. M., Alahi, A., Fleuret, F., Fua, P., and Savarese, S. (2017). Social Scene Understanding: End-to-End Multi-person Action Localization and Collective Activity Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3425–3434. 8, 38, 40

Behrmann, N., Gall, J., and Noroozi, M. (2021). Unsupervised video representation learning by bidirectional feature prediction. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1670–1679. 10

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166. 17

Biswas, S. and Gall, J. (2018). Structural Recurrent Neural Network (SRNN) for Group Activity Analysis. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1625–1632. 29

Biswas, S. and Gall, J. (2020). Discovering Multi-Label Actor-Action Association in a Weakly Supervised Setting. In *Asian Conference on Computer Vision (ACCV)*, pages 547–561. 51

Biswas, S. and Gall, J. (2021). Multiple Instance Triplet Loss for Weakly Supervised Multi-Label Action Localisation of Interacting Persons. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 2159–2167. 77

Biswas, S., Souri, Y., and Gall, J. (2019). Hierarchical Graph-RNNs for Action Detection of Multiple Activities. In *IEEE International Conference on Image Processing (ICIP)*, pages 1–5. 43

Bojanowski, P., Bach, F., Laptev, I., Ponce, J., Schmid, C., and Sivic, J. (2013). Finding actors and actions in movies. In *International Conference on Computer Vision (ICCV)*, pages 2280–2287. 9

Briggs, F., Fern, X. Z., and Raich, R. (2012). Rank-loss support instance machines for MIML instance annotation. In *Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD)*, pages 534–542. 10

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, pages 213–229. Springer. 9

Carreira, J. and Zisserman, A. (2017). Quo Vadis, Action recognition? A new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733. v, 2, 15, 20, 21, 30, 44, 45, 48, 55, 57, 65, 93

Castellano, B. (2018). PySceneDetect. `https://github.com/Breakthrough/PySceneDetect`. 65

Chao, Y.-W., Vijayanarasimhan, S., Seybold, B., Ross, D. A., Deng, J., and Sukthankar, R. (2018). Rethinking the Faster R-CNN Architecture for Temporal Action Localization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1130–1139. 8

Chappa, N. V., Nguyen, P., Nelson, A. H., Seo, H.-S., Li, X., Dobbs, P. D., and Luu, K. (2023). Spartan: Self-supervised spatiotemporal transformers approach to group activity recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5157–5167. 8, 92

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, pages 1597–1607. 10, 81

Chéron, G., Alayrac, J.-B., Laptev, I., and Schmid, C. (2018). A flexible model for training action localization with varying levels of supervision. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 942–953. 52, 53, 55

Cho, K., van Merriënboer, B., Gulçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. 18, 19

Choi, W. and Savarese, S. (2014). Understanding Collective Activities of people from videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(6):1242–1257. 7

Choi, W., Shahid, K., and Savarese, S. (2009). What are they doing?: Collective activity classification using spatio-temporal relationship among people. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1282–1289. 7

Choi, W., Shahid, K., and Savarese, S. (2011). Learning context for collective activity recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3273–3280. 7

Dantzig, G. B. (1990). Origins of the simplex method. In *A history of scientific computing*, pages 141–151. 26

Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., and Adam, H. (2014). Large-scale object classification using label relation graphs. In *European Conference on Computer Vision (ECCV)*, pages 48–64. 58

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. 21

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019). Arcface: Additive angular margin loss for deep face recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699. 10

Deng, Z., Vahdat, A., Hu, H., and Mori, G. (2016). Structure Inference Machines: Recurrent Neural Networks for Analyzing Relations in Group Activity Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4772–4781. 7, 30

Deng, Z., Zhai, M., Chen, L., Liu, Y., Muralidharan, S., Roshtkhari, M. J., and Mori, G. (2015). Deep Structured Models for Group Activity Recognition. In *The British Machine Vision Conference (BMVC)*, pages 179.1–179.12. 8

Derrington, A. and Lennie, P. (1984). Spatial and temporal contrast sensitivities of neurones in lateral geniculate nucleus of macaque. *The Journal of Physiology*, 357(1):219–240. 22

Diba, A., Fayyaz, M., Sharma, V., Paluri, M., Gall, J., Stiefelhagen, R., and Van Gool, L. (2020). Large scale holistic video understanding. In *European Conference on Computer Vision (ECCV)*, pages 593–610. 79

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in neural information processing systems (NeurIPS)*, pages 2224–2232. 9

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., and Feichtenhofer, C. (2021). Multiscale vision transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6824–6835. 9, 74

Feichtenhofer, C. (2020). X3d: Expanding architectures for efficient video recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 203–213. 74

Feichtenhofer, C., Fan, H., Malik, J., and He, K. (2019). Slowfast Networks for Video Recognition. In *International Conference on Computer Vision (ICCV)*, pages 6202–6211. v, 9, 20, 22, 52, 55, 57, 65, 70, 74, 79

Feng, L., Tung, F., Ahmed, M. O., Bengio, Y., and Hajimirsadegh, H. (2024). Were rnns all we needed? *arXiv preprint arXiv:2410.01201*. 18

Ghadiyaram, D., Tran, D., and Mahajan, D. (2019). Large-scale weakly-supervised pre-training for video action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12046–12055. 9

Gilmore, P. C. and Gomory, R. E. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, 9(6):849–859. 27

Gilmore, P. C. and Gomory, R. E. (1963). A linear programming approach to the cutting stock problem—part ii. *Operations Research*, 11(6):863–888. 27

Girdhar, R., Carreira, J., Doersch, C., and Zisserman, A. (2018). A better baseline for AVA. *CoRR*, abs/1807.10066. 8, 44, 49, 50

Girdhar, R., Carreira, J., Doersch, C., and Zisserman, A. (2019). Video Action Transformer Network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 244–253. 52, 55, 72, 74, 79, 81

Girshick, R. (2015). Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, pages 1440–1448. v, 8, 23, 24

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587. 23

Gkioxari, G. and Malik, J. (2015). Finding action tubes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 759–768. 8, 52, 55

Goodale, M. A. and Milner, A. D. (1992). Separate visual pathways for perception and action. *Trends in Neurosciences*, 15(1):20–25. 20

Gori, M., Monfardini, G., and Scarselli, F. (2005). A New Model for Learning in Graph Domains. In *International Joint Conference on Neural Networks (IJCNN)*, pages 729–734. 44, 47

Gritsenko, A. A., Xiong, X., Djolonga, J., Dehghani, M., Sun, C., Lucic, M., Schmid, C., and Arnab, A. (2024). End-to-end spatio-temporal action localisation with video transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18373–18383. 9

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6047–6056. v, 5, 9, 44, 45, 46, 48, 49, 50, 54, 64, 65, 78, 79, 80, 81, 82, 85

Gupta, T., Vahdat, A., Chechik, G., Yang, X., Kautz, J., and Hoiem, D. (2020). Contrastive learning for weakly supervised phrase grounding. In *European Conference on Computer Vision (ECCV)*, pages 752–768. 10

Hansen, T. D., Miltersen, P. B., and Zwick, U. (2013). Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM*, 60(1):1–16. 26

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738. 10

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778. 45

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780. 18, 37

Hoffer, E. and Ailon, N. (2015). Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer. 10

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366. 13

Hou, R., Chen, C., and Shah, M. (2017). Tube Convolutional Neural Network (T-CNN) for Action Detection in Videos. In *International Conference on Computer Vision (ICCV)*, pages 5822–5831. 8, 52

Ibrahim, M. S., Muralidharan, S., Deng, Z., Vahdat, A., and Mori, G. (2016a). A Hierarchical Deep Temporal Model for Group Activity Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1971–1980. 7, 9, 30, 33, 35, 37, 38

Ibrahim, M. S., Muralidharan, S., Deng, Z., Vahdat, A., and Mori, G. (2016b). Hierarchical Deep Temporal Models for Group Activity Recognition. *CoRR*, abs/1607.02643. vi, xi, 30, 37, 38, 40

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014). Human3.6M: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(7):1325–1339. 8

Jain, A., Zamir, A. R., Savarese, S., and Saxena, A. (2016). Structural-RNN: Deep Learning on Spatio-Temporal Graphs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5308–5317. 8, 9, 31, 34

Jhuang, H., Gall, J., Zuffi, S., Schmid, C., and Black, M. J. (2013). Towards understanding action recognition. In *International Conference on Computer Vision (ICCV)*, pages 3192–3199. 52, 79

Ji, J., Krishna, R., Fei-Fei, L., and Niebles, J. C. (2020). Action genome: Actions as compositions of spatio-temporal scene graphs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10236–10247. 79

Ji, S., Xu, W., Yang, M., and Yu, K. (2013). 3D Convolutional Neural Networks for Human Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(1):221–231. 15, 30

Jiang, Y., Cui, K., Chen, L., Wang, C., and Xu, C. (2020). Soccerdb: A large-scale database for comprehensive video understanding. In *International Workshop on Multimedia Content Analysis in Sports*, pages 1–8. v, 3

Kalogeiton, V., Weinzaepfel, P., Ferrari, V., and Schmid, C. (2017). Action Tubelet Detector for spatio-temporal action localization. In *International Conference on Computer Vision (ICCV)*, pages 4415–4423. 8, 52

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. In *ACM Symposium on Theory of computing*, pages 302–311. 26

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Li, F. (2014). Large-Scale Video Classification with Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732. 30

Kerrigan, A., Duarte, K., Rawat, Y., and Shah, M. (2021). Reformulating zero-shot action recognition for multi-label actions. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:25566–25577. 93, 94

Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466. 15

King, D. E. (2009). Dlib-ml: A machine Learning Toolkit. *JMRL*, 10:1755–1758. 37

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*. 9

Koppula, H. S., Gupta, R., and Saxena, A. (2013). Learning human activities and object affordances from RGB-D videos. *International Journal of Robotics Research*, 32(8):951–970. 8

Koppula, H. S. and Saxena, A. (2016). Anticipating Human Activities using Object Affordances for Reactive Robotic Response. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(1):14–29. 8

Kovtun, E., Boeva, G., Zabolotnyi, A., Burnaev, E., Spindler, M., and Zaytsev, A. (2023). Label attention network for sequential multi-label classification: you were looking at a wrong self-attention. *CoRR*, abs/2303.00280. 94

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Conference on Neural Information Processing Systems (NeurIPS)*, 25:1097–1105. 15, 20, 37

Kühne, H., Richard, A., and Gall, J. (2018). A hybrid RNN-HMM approach for weakly supervised temporal action segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 765–779. 4, 9

Kukleva, A., Tapaswi, M., and Laptev, I. (2020). Learning interactions and relationships between movie characters. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9849–9858. 93

Lai, Q., Zhou, J., Gan, Y., Vong, C.-M., and Chen, C. P. (2023). Single-stage broad multi-instance multi-label learning (bmiml) with diverse inter-correlations and its application to medical image classification. *IEEE Transactions on Emerging Topics in Computational Intelligence*. 10

Laptev, I., Marszalek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. 9

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. v, 15, 16

Lee, Y. T. and Sidford, A. (2015). Efficient inverse maintenance and faster algorithms for linear programming. In *Annual Symposium on Foundations of Computer Science*, pages 230–249. IEEE. 26, 27

Li, D., Qiu, Z., Dai, Q., Yao, T., and Mei, T. (2018). Recurrent tubelet proposal and recognition networks for action detection. In *European Conference on Computer Vision (ECCV)*, pages 303–318. 8

Li, J., Liu, J., Yongkang, W., Nishimura, S., and Kankanhalli, M. (2020). Weakly-supervised multi-person action recognition in 360° videos. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 508–516. 9, 10

Li, S., Cao, Q., Liu, L., Yang, K., Liu, S., Hou, J., and Yi, S. (2021). Groupformer: Group activity recognition with clustered spatial-temporal transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13668–13677. 8, 9, 92

Li, Y., Zemel, R., Brockschmidt, M., and Tarlow, D. (2016). Gated Graph Sequence Neural Networks. In *International Conference on Learning Representations (ICLR)*. 9, 44, 47

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017). Focal Loss for Dense Object Detection. In *International Conference on Computer Vision (ICCV)*, pages 2980–2988. 47, 48

Liu, X., Li, H., Shao, J., Chen, D., and Wang, X. (2018). Show, Tell and Discriminate: Image Captioning by Self-retrieval with Partially Labeled Data. In *European Conference on Computer Vision (ECCV)*, pages 338–354. 10

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *International Conference on Computer Vision (ICCV)*, pages 10012–10022. 20

Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., and Hu, H. (2022). Video swin transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3202–3211. 20

Martin, E. and Cundy, C. (2018). Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations (ICLR)*. 18

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133. 12

Mettes, P., Snoek, C. G., and Chang, S.-F. (2017). Localizing actions from video labels and pseudo-annotations. In *The British Machine Vision Conference (BMVC)*. 52, 53, 55

Min, K. (2023). Sthg: Spatial-temporal heterogeneous graph learning for advanced audio-visual diarization. *CoRR*, abs/2306.10608. 9

Min, K., Roy, S., Tripathi, S., Guha, T., and Majumdar, S. (2022). Learning long-term spatial-temporal graphs for active speaker detection. In *European Conference on Computer Vision (ECCV)*, pages 371–387. 9

Mitra, G. (1973). Investigation of some branch and bound strategies for the solution of mixed integer linear programs. *Mathematical Programming*, 4:155–170. 27

Mkhallati, H., Cioppa, A., Giancola, S., Ghanem, B., and Van Droogenbroeck, M. (2023). Soccernet-caption: Dense video captioning for soccer broadcasts commentaries. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5073–5084. 93

Nguyen, C.-T., Zhan, D.-C., and Zhou, Z.-H. (2013). Multi-modal image annotation with multi-instance multi-label LDA. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1558–1564. 10, 84

Nguyen, N. (2010). A new SVM approach to multi-instance multi-label learning. In *IEEE International Conference on Data Mining (ICDM)*, pages 384–392. 10, 84

Oh Song, H., Xiang, Y., Jegelka, S., and Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4004–4012. 10

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748. 10

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. (2023). Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning (ICML)*, pages 26670–26698. PMLR. 18

Pan, J., Chen, S., Shou, M. Z., Liu, Y., Shao, J., and Li, H. (2021). Actor-context-actor relation network for spatio-temporal action localization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 464–474. 9, 55

Qi, S., Wang, W., Jia, B., Shen, J., and Zhu, S.-C. (2018). Learning Human-Object interactions by Graph Parsing Neural Networks. In *European Conference on Computer Vision (ECCV)*, pages 401–417. 9

Qi, X., Liao, R., Jia, J., Fidler, S., and Urtasun, R. (2017). 3D Graph Neural Networks for RGBD Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5199–5208. 9

Qian, R., Meng, T., Gong, B., Yang, M.-H., Wang, H., Belongie, S., and Cui, Y. (2021). Spatiotemporal contrastive video representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6964–6974. 10

Rajasegaran, J., Pavlakos, G., Kanazawa, A., Feichtenhofer, C., and Malik, J. (2023). On the benefits of 3d pose and tracking for human action recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 640–649. 9

Ramanathan, V., Huang, J., Abu-El-Haija, S., Gorban, A. N., Murphy, K., and Fei-Fei, L. (2016). Detecting Events and Key Actors in Multi-person Videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3043–3053. 9, 30

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. 23

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in neural information processing systems (NeurIPS)*, pages 91–99. vi, 23, 24, 45, 65, 81, 85

Richard, A., Kühne, H., and Gall, J. (2018). Action sets: Weakly supervised action segmentation without ordering constraints. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5996. 4, 9, 20

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407. 15

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. 12

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536. 12, 15

Ryoo, M. S. and Aggarwal, J. K. (2009). Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *International Conference on Computer Vision (ICCV)*, pages 1593–1600. 7

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80. 9, 44, 47

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823. 10

Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: a local svm approach. In *International Conference on Pattern Recognition (ICPR)*, pages 32–36. v, 2

Shah, K., Shah, A., Lau, C. P., de Melo, C. M., and Chellappa, R. (2023). Multi-view action recognition using contrastive learning. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3381–3391. 10

Shu, T., Todorovic, S., and Zhu, S. (2017). CERN: Confidence-Energy Recurrent Network for Group Activity Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4255–4263. 7, 30, 33, 37, 38, 40

Sigurdsson, G. A., Varol, G., Wang, X., Farhadi, A., Laptev, I., and Gupta, A. (2016). Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding. In *European Conference on Computer Vision (ECCV)*, pages 510–526. 79

Simonyan, K. and Zisserman, A. (2014a). Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances in Neural Information Processing Systems*, pages 568–576. v, 20, 21, 30

Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556. 15, 20, 37

Singh, B., Marks, T. K., Jones, M. J., Tuzel, O., and Shao, M. (2016). A Multi-stream Bi-directional Recurrent Neural Network for fine-grained action detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1961–1970. 30

Singh, G., Saha, S., Sapienza, M., Torr, P., and Cuzzolin, F. (2017). Online Real-Time multiple spatiotemporal action localisation and prediction. In *International Conference on Computer Vision (ICCV)*, pages 3657–3666. 8, 52

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems (NeurIPS)*, pages 1857–1865. 10

Soomro, K. and Shah, M. (2017). Unsupervised action discovery and localization in videos. In *International Conference on Computer Vision (ICCV)*, pages 696–705. 52, 53, 55

Soomro, K., Zamir, A. R., and Shah, M. (2012). Ucf101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402. 52

Stroud, J. C., Ross, D. A., Sun, C., Deng, J., and Sukthankar, R. (2018). D3d: Distilled 3d networks for video action recognition. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 625–634. 44, 49, 50

Sun, C., Shrivastava, A., Vondrick, C., Murphy, K., Sukthankar, R., and Schmid, C. (2018). Actor-Centric Relation Network. In *European Conference on Computer Vision (ECCV)*, pages 318–334. 44, 49, 50, 52, 72

Sun, C., Shrivastava, A., Vondrick, C., Sukthankar, R., Murphy, K., and Schmid, C. (2019). Relational action forecasting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 273–283. 52, 55, 72

Sun, Y., Cheng, C., Zhang, Y., Zhang, C., Zheng, L., Wang, Z., and Wei, Y. (2020). Circle loss: A unified perspective of pair similarity optimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6398–6407. 10

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. 40

Tamura, M., Vishwakarma, R., and Vennelakanti, R. (2022). Hunting group clues with transformers for social group activity recognition. In *European Conference on Computer Vision (ECCV)*, pages 19–35. 8, 9

Tang, J., Xia, J., Mu, X., Pang, B., and Lu, C. (2020). Asynchronous Interaction Aggregation for Action Detection. In *European Conference on Computer Vision (ECCV)*, pages 71–87. 74

Tang, Y., Wang, Z., Li, P., Lu, J., Yang, M., and Zhou, J. (2018). Mining Semantics-Preserving Attention for Group Activity Recognition. In *ACM International Conference on Multimedia (ACMMM)*, pages 1283–1291. 9

Teney, D., Liu, L., and van den Hengel, A. (2017). Graph-Structured Representations for Visual Question Answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3233–3241. 9

Ulutan, O., Rallapalli, S., Srivatsa, M., Torres, C., and Manjunath, B. S. (2020). Actor conditioned attention maps for video action detection. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 527–536. 9

Van Essen, D. C. and Gallant, J. L. (1994). Neural mechanisms of form and motion processing in the primate visual system. *Neuron*, 13(1):1–10. 22

Vicol, P., Tapaswi, M., Castrejon, L., and Fidler, S. (2018). Moviegraphs: Towards understanding human-centric situations from videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8581–8590. 20, 93

Wang, J., Zhou, F., Wen, S., Liu, X., and Lin, Y. (2017). Deep metric learning with angular loss. In *International Conference on Computer Vision (ICCV)*, pages 2593–2601. 10

Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. (2019). Multi-similarity loss with general pair weighting for deep metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5022–5030. 10

Weinzaepfel, P., Martin, X., and Schmid, C. (2016). Towards weakly-supervised action localization. *CoRR*, abs/1605.05197. 8

Wen, Y., Liu, M., Wu, S., and Ding, B. (2024). Chase: Learning convex hull adaptive shift for skeleton-based multi-entity action recognition. In *Conference on Neural Information Processing Systems (NeurIPS)*. 8

Wu, C.-Y., Feichtenhofer, C., Fan, H., He, K., Krahenbuhl, P., and Girshick, R. (2019). Long-term feature banks for detailed video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 284–293. 52, 55, 74

Wu, C.-Y. and Krahenbuhl, P. (2021). Towards Long-Form Video Understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1884–1894. 74

Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. (2017). Sampling matters in deep embedding learning. In *International Conference on Computer Vision (ICCV)*, pages 2840–2848. 10

Wu, Y., Zhu, L., Wang, X., Yang, Y., and Wu, F. (2020). Learning to anticipate egocentric actions by imagination. *IEEE Transactions on Image Processing (IEEE TIP)*, 30:1143–1152. 10

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 55, 65, 67, 71, 81, 85

Yang, H., Tianyi Zhou, J., Cai, J., and Soon Ong, Y. (2017). MIML-FCN+: Multi-instance multi-label learning via fully convolutional networks with privileged information. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1577–1585. 10, 54, 55, 84

Yang, J., Lu, J., Lee, S., Batra, D., and Parikh, D. (2018). Graph R-CNN for Scene Graph Generation. In *European Conference on Computer Vision (ECCV)*, pages 670–685. 9

Yuan, H., Ni, D., and Wang, M. (2021). Spatio-temporal dynamic inference network for group activity recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7476–7485. 8, 92

Zappardino, F., Uricchio, T., Seidenari, L., and Del Bimbo, A. (2021). Learning group activities from skeletons without individual action labels. In *International Conference on Pattern Recognition (ICPR)*, pages 10412–10417. 92

Zha, Z.-J., Hua, X.-S., Mei, T., Wang, J., Qi, G.-J., and Wang, Z. (2008). Joint multi-label multi-instance learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. 10

Zhang, X.-Y., Shi, H., Li, C., and Li, P. (2020). Multi-instance multi-label action recognition and localization based on spatio-temporal pre-trimming for untrimmed videos. In *Annual AAAI Conference on Artificial Intelligence (AAAI)*, pages 12886–12893. 10

Zhang, Y.-H., Xiao, J., Yang, S., and Shan, S. (2019). Multi-task learning for audio-visual active speaker detection. *The ActivityNet Large-Scale Activity Recognition Challenge*, 4. 94

Zhou, H., Kadav, A., Shamsian, A., Geng, S., Lai, F., Zhao, L., Liu, T., Kapadia, M., and Graf, H. P. (2022). Composer: compositional reasoning of group activity in videos with keypoint-only modality. In *European Conference on Computer Vision (ECCV)*, pages 249–266. 8

Zhou, L., Xu, C., and Corso, J. (2018). Towards automatic learning of procedures from web instructional videos. In *Annual AAAI Conference on Artificial Intelligence (AAAI)*, pages 7590—-7598. v, 2

Zhou, Z.-H. and Zhang, M.-L. (2006). Multi-instance multi-label learning with application to scene classification. In *Advances in neural information processing systems (NeurIPS)*, pages 1609–1616. 10, 54, 55

Zhou, Z.-H., Zhang, M.-L., Huang, S.-J., and Li, Y.-F. (2012). Multi-instance multi-label learning. *Artificial Intelligence*, 176:2291–2320. 10, 54, 55