University of Perugia

Department of Engineering

Degree Course in

Ingegneria Informatica e Robotica

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)

Master's Thesis in

Analysing social interactions through a wearable camera: a first-person point of view

Supervisor
Gabriele Costante

Advisor
Mariella Dimiccoli

Author
Simone Felicioni

Academic Year 2018-2019

Dedicated to my family
for your support,
for your love.

Abstract

Nowadays social interaction analysis in egocentric vision is an increasingly popular topic due to its wide range of potential applications in domains including social robotics, social care and assistive technology. This Thesis aims to develop an automatic social interaction classification approach for egocentric videos captured by a wearable camera. The proposed method consists of four major steps. The first step is the extraction of social cues in videos. The second step is the construction of a graph based on the extracted features, where people in the scene can be seen as nodes and relations between them as edges. Then, a Relational Graph Convolutional Network is used on the graph to extract feature at the frame level. The third step is based on a gated recurrent unit to extract context information from the sequence of frames. The last step is the classification of the social interactions seen in the scene. Performances of this approach are evaluated with the Fathi et al. [1] first-person social interaction dataset. The current state of the art had never tried to deal with this problem using graph based neural networks, but experimental results show that this method is a valid mean to classify social interactions in egocentric video.

Contents

1 Introduction **6**

2 Related work **10**

2.1 Social interaction analysis 10

2.1.1 Third-person perspective 10

2.1.2 Egocentric perspective 12

2.2 Graph-based analysis 15

3 Proposed approach **19**

3.1 Problem formulation 20

3.2 Feature extraction 21

3.2.1 Microsoft Azure Cognitive Service 21

3.2.2 Distance from camera 23

3.2.3 Pairwise Distance 25

3.2.4 Mutual Attention 25

3.2.4.1 Notation 26

3.2.4.2 Algorithm 26

3.2.5 First-Person Head Motion 27

3.2.5.1 Optical Flow 28

3.2.5.2 Homography estimation 29

3.3 Frame level modelling via R-GCN 30
3.3.1 Notation 30
3.3.2 Node embeddings 31
3.3.3 The basic Graph Neural Network 32
3.3.4 Graph Convolutional Neural Network 34
3.3.5 Relational Graph Convolutional Neural Network 37
3.4 Temporal modelling via GRU 39
3.4.1 Gated Recurrent Unit 41
3.5 Classification step 44
3.6 The final model 45

4 Experimental Results **48**
4.1 Dataset 48
4.2 Implementation details 50
4.2.1 Pre-processing 50
4.2.2 Feature scaling 53
4.2.3 Data augmentation 53
4.2.4 Hyperparameters tuning 53
4.2.5 Graph construction 54
4.2.6 Experimental settings 54
4.3 Results 56
4.4 Ablation study 57
4.5 Discussion 58

5 Conclusions **61**

References **70**

List of Figures

1.1 Workflow of our method 9

3.1 People grouping: in different frames, people can be grouped by comparing faces and a unique ID is assigned for each person . . . 22
3.2 3D head orientation 23
3.3 Polynomial regression for distance estimation 24
3.4 Pairwise distance estimation 25
3.5 Bird-View Model for mutual attention definition 27
3.6 Examples of motion features extracted from walking and not-walking interactions 29
3.7 2D convolution vs graph convolution [2] 34
3.8 Graph Neural Network architecture from Kipf et al. 2016 36
3.9 Diagram for nodes update in R-GCN model 38
3.10 An unrolled recurrent neural network 40
3.11 Description of a RNN block 41
3.12 Gated Recurrent Unit 42
3.13 Example of concat pooling method 44
3.14 The final model used in this project 47

4.1 Histogram of label distribution 49

4.2 Example of video clips belonging to different conversation types in the dataset 50
4.3 Confusion matrix obtained with our model 59

List of Tables

4.1 Performance comparison with the SOTA and other baselines. . . . 56
4.2 Performance comparison by removing a feature at time. 57
4.3 Per class classification recall comparison by removing a feature at time. 58

Chapter 1

Introduction

The recent trend in the Computer Vision community is shifting the attention from a mere pattern recognition perspective to a high-level scene understanding, looking for a description and interpretation of the scene. This analysis is gaining increasing attention thanks to the wide range of applications and it needs a deep understanding of social signals between people, for example to identify relations and interactions. This new perspective is based on principles from the social and psychological literature and it is called Social Signal Processing (SSP) [3]. SSP takes advantage mainly of nonverbal cues such as face expressions, gaze direction, body posture, relative distances in the space and so on. In video surveillance, groups of people are present and behaviour analysis is more and more important to understand social meaning of actions for security reasons [4]. Thanks to a fast development of wearable devices, cameras can be worn by a subject and they record high-quality videos providing an intimate perspective and a natural visual experience in an unconstrained setting. For this reason, first-person videos present less occlusion compared to fixed-camera videos because people naturally move to provide a clear view of people they are interacting with. So, egocentric vision system is an interesting choice since it allows to capture social cues and it

can be applied to fields as assistive technology, human-human and human-robot interaction, where a robot is asked to have social skills while assisting people. Most of the works on social events analysis aim to detect if a social interaction is present or not by using the number of people in the scene, gaze direction and other social cues. So far just a few works have addressed the problem of social interaction classification in terms of conversational type, e.g. discussion or monologue [1].

The main objective of this project is to extend the state of the art in this latter research area by developing an automatic social interaction classification approach from videos captured by a wearable camera. Our approach is taking advantage of social network structure of interactions in social events by using a graph based neural network. It is an end-to-end recent technique that allows to learn a representation of graph structured data and to fit a predictive model on them. Graph Neural Networks are a reasonable choice for our project because they model complex relationship between real-world data that can be represented in a graph and they have a strong representation learning capability.

Following the literature in social interaction classification, interactions are categorized into five types: dialogue, walk dialogue, discussion, walk discussion and monologue. Discussion means that multiple people are involved in the interaction and it is interactive, dialogue that only two people are involved and monologue that it is one-sided. This method uses faces and first-person motion as main sources of information for the classification. The workflow is shown in Figure 1.1: first of all, both relational and non-relational features are extracted from raw videos. A graph can be built by using relational features for edges and non-relational ones for nodes. Then, a Relational Graph Convolutional Neural Network is used to extract an embedding for each graph and its output is concatenated with first-person motion features. A Gated Recurrent Unit extracts

sequential context information from the frame sequence, taking into account the evolution of the features over time. Finally, a classification step is used to identify the class to whom the video clip belongs.

Videos in the dataset had been recorded by Fathi et al. in [1] and available online[1]. This dataset contains day-long egocentric videos recording social events in an amusement park.

This Thesis is divided into five main chapters to explain as well as possible the methods and the choices made to reach the final experimental results.

- Chapter 2 gives an overview of the State of the Art on the analysis of social interactions from both third and first-person perspective.
- Chapter 3 presents an overview of the proposed approach and architecture. The various sections aim to go deeper in the used methods to justify the choices, illustrating theory and practical application.
- Chapter 4 presents the implementation details and the experimental results with an interpretation and evaluation of them. Graphs of training/validation curves, confusion matrices and test results are shown and results by using different features are compared.
- Chapter 5 finally draws conclusions on this project.

This project was developed in the Perception and Manipulation Laboratory of the Industrial Informatics and Robotics Institute in Barcelona during an Erasmus+ internship. The code has been totally developed in Python 3.6 using PyTorch framework in a Ubuntu 18.04 environment.

[1] `http://ai.stanford.edu/~alireza/Disney/`

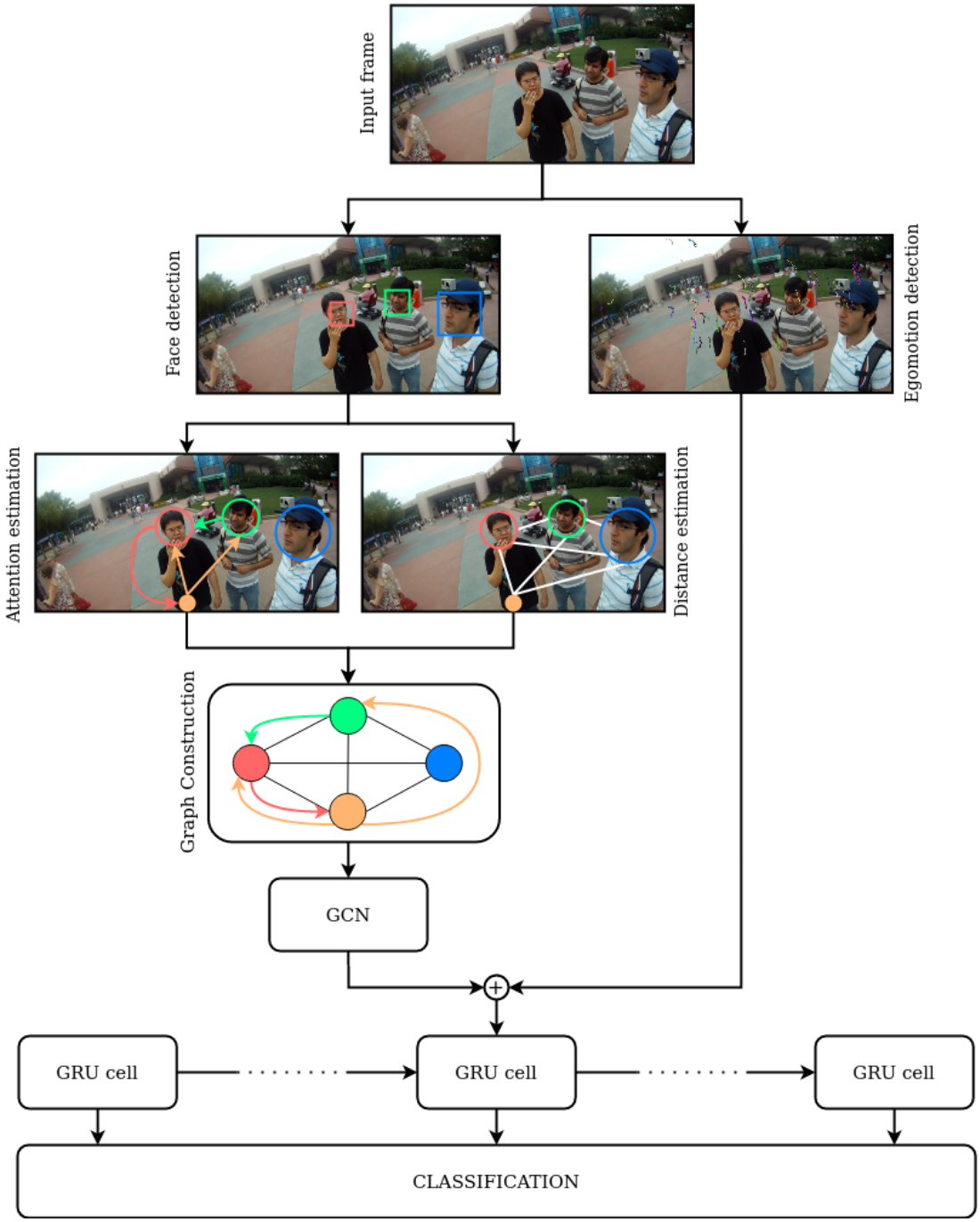

Figure 1.1: Workflow of our method

Chapter 2

Related work

2.1 Social interaction analysis

In the last years, many works aimed to analyse social roles, interactions and social events in general. It has proved to be a challenging task due to the fact that this analysis relies on non-verbal cues which have to be identified. These events can be explored with third-person (fixed-camera) or first-person (egocentric) perspective. Both of them present different issues, such as occlusion in fixed-camera videos or unpredictable movements of the camera wearer in egocentric ones. The just mentioned non-verbal cues may change depending on the task: facial expressions, relative position, mutual gaze or body posture could be useful for a high-level scene understanding.

2.1.1 Third-person perspective

Third-person images and videos were widely used for social events analysis. Most of the works dealt with problem of recognizing family relations in photo albums by using relative position and pose, attributes such as edge and gender of each person ([5], [6], [7], [8], [9], [10], [11], [12]). Most of these works focused on family

relationship, such as parents-children, grandparents-grandchildren or husband-wife, looking for visual patterns in these kind of relations.

Social roles were studied in many works ([13], [14], [15], [16]) by using relative age, gender, clothing and location of people in the image. Roles can be interpreted as relations among participants of a social event or permanent interpersonal relations. For example, immediate social roles can be studied in a birthday party, where there are the birthday child, the parents and the guests. On the other hand, an example of permanent role can be the leader-subordinate relation.

Sun et al. in [17] proposed a method to predict social domain and relation from image data based on Bugental's domain-based theory [18], which divides into five domains and sixteen related sub-categories. In particular, social domain theory says that 5 domains cover all relevant aspects of people interactions and those domains manifest themselves in concrete behaviours. For example, the *attachment domain* is characterized by protective proximity (e.g. parents with children), the *reciprocity domain* is based on equality on individuals key features, the *mating domain* is for romantic relationship, and so on. Sun et al. collected a PIPA (People in Photo Album) dataset containing 37107 Flickr photos and experimented with two types of models. The first model is an end-to-end CNN model where a pair of body regions is mapped into the relation classes. The second model trains CNN for semantic attributes derived from social domain theory, then uses the concatenated features for the learning phase through linear SVM. In particular, the attributed used are age, gender, location, head appearance, head pose, face emotion, clothing, proximity and activity.

Cristani et al. in [4] studied how social signals and nonverbal cues, e.g. face expression, mutual gaze, body posture and relative distance, may improve the human behavior analysis. Firstly, they present a short review of classical systems

for video-surveillance defining a list of problems without a social analysis (e.g. definition of threatening behavior, modeling of groups and interactions). Then, they proposed SSP to codify human behaviors by analysing different aspects such as physical appearance, gesture, posture, gaze, vocal behavior and the space. Most of them cannot be used in a typical surveillance scenario, but cues related to the environment appear particularly important for the interpersonal distance (that is associated with different degrees of intimacy) and spatial arrangements of interacting people (to separate the group of interacting individuals to the others). Finally, they collected different SOTA approaches which exploit spatial information for surveillance purposes.

Liu et al. in [19] proposed a graph based method to classify social relations from videos, using the eight more common subcategories among the ones proposed in [17]. Firstly, they extracted global features by using a Temporal Segment Network (TSN). Then, they combined with a Triple Graphs method that crops people and objects as region of interests with a pre-trained Mask R-CNN to build intra-person, inter-person and person-object graphs. Finally, they proposed a Pyramid Graph Convolution Network (PGCN) able to learn both long-term and short-term information with a pyramid of temporal receptive fields.

2.1.2 Egocentric perspective

First-person videos present different challenges in comparison to fixed cameras, for example for the variation of the viewpoint and for the focus of attention, that may not always be in the wearable camera's field of view. But analysis of egocentric videos have gained interest of researchers for different types of tasks, such as object recognition ([20], [21]), activity recognition [22] and person-object interaction recognition ([22], [23]). Other works aimed to detect groups of interacting individuals ([24], [25], [26]), social saliency ([27], [28]), social relations [29]

or who is interacting with the camera wearer [30].

Alletto et al. in [24] provided a social interactions detection approach by using head mounted cameras. Their method is based on people's head pose combining landmarks and shape descriptors, 3D localization and mutual distance estimation. Tracked faces and the latter components are used to build a bird-view model, that will be the input of the supervised correlation clustering algorithm using structural SVM in order to detect social groups based on the estimation of pairwise relations of their members. First of all, they decided which frame is worth to elaborate, because head motion can cause a significant blur in the video sequence and so a low-quality frame. To deal with this typical challenge of the ego-vision scenario, the amount of blurriness in each frame is evaluated to decide whether to proceed with the tracking or to skip it. Secondly, to estimate face pose they used two different techniques: face landmarks and shape based estimation. The first approach allows an accurate estimation because the face resolution is high enough, but if the landmark estimation fails, this method uses HOG features and a classification framework composed of SVM followed by HMM. Then, the relationship between two people p and q is given by their mutual distance d and their face rotation o_{pq} and o_{qp} (with $o_{pq} = o_{qp} = 0$ if p and q are facing each other). As just mentioned, to partition social groups based on the pairwise relations of their members, they applied the correlation clustering algorithm based on an affinity matrix W where W_{pq} means the probability for p and q to belong to the same group.

Aghaei et al. in [30] developed a LSTM-based method to detect the moments when the camera wearer is interacting exploiting distance and orientation of the individuals in the photostreams. This method first estimates discretized head orientation, people localization in the 2D image and distance from the camera through a polynomial regression. To analyse temporal change they introduced

feature vectors as input for LSTM to classify sequences of different lengths as interaction or not. Moreover, Aghaei et al. in [31] detected social interactions in egocentric videos by using social signals, e.g. distance, head orientation and face expression, through LSTM. The second task is to categorize these interactions into different social meetings, i.e. formal and informal meetings, based on environmental features and face expressions, through a second LSTM. To extract global features, they proposed to quantize CNN features re-writing them as discrete words and then apply PCA to keep the most important components of the quantization result. Moreover, they showed bar-plot of eight facial expressions using ground truth information, and these bar-plots suggested that people express more freely their emotions in informal meetings. Finally, social patterns are characterized by using four concepts, i.e. frequency, social trend, diversity and duration. This analysis implies the ability of defining the nature of social interactions of the user from various temporal and social aspects.

S. Bano et al. in [32] worked in social interaction detection where the camera wearer is involved in focused interaction. A focused interaction occurs when two or more individuals interact establishing face-to-face engagement and direct conversation. They used a HOG-based face detector in each frame and KLT point tracking to refine face detection results. For Voice Activity Detection (VAD) they combined four types of discriminative audio features to detect voice activity in noisy environments, that are spectral shape, spectro-temporal modulations, harmonicity and long-term spectral variability. The result is a VAD score range from 0 (i.e. no voice activity) to 1 (i.e. high confidence for voice activity). In order to fuse audio and visual features, that are obtained at different sampling rates, the audio stream was down-sampled and features were normalised to have zero-mean and unit variance. They compared results by using LSTM neural network or SVMs and by using audio-only, video-only or audio-visual features.

Fathi et al. in [1] were the first to classify social interaction in egocentric videos and they collected the dataset used in this project. They chose first-person video because thanks to their several advantages in comparison to fixed video recorders: for example a wearable camera records what the camera wearer is attending and occlusion is less common because he naturally moves to have a clearer view. They mainly used two sources of information to analyze the scene: faces and first-person motion. In particular, social interactions are characterized by the pattern of attention shift and turn-taking over time. Firstly, they detected and tracked faces belonging to different interacting individuals, then they used 3D localization and orientation to aproximate line of sight. For this purpose, they assumed that it is more likely that a person is looking at an other person than an object. They built an MRF model to estimate where each person is looking at, and it is able to realize if people are looking at someone whose face is not detected. Then, first-person head movement provides significant information in cases where two individuals are speaking while walking, so that faces cannot be seen in the video. They modelled this problem by using a Hidden Conditional Random Field (HCRF), where frames are assigned hidden state labels and these states are connected by a chain over time. Our goal is to verify if our framework works better than SOTA method. So, we are going to compare our results with theirs by comparing our confusion matrix result with the one reported in [1].

2.2 Graph-based analysis

An image can be represented as a regular grid in the Euclidean space, so that a convolutional neural network (CNN) is able to extract local meaningful features for image analysis. In general, the strength in the CNNs and RNNs is in the capability of exploiting the interconnections between input data, that are spatial

connection for CNNs and temporal connection for RNNs. These data can be seen as perfectly regular structured graphs: both pixel in an image and samples in a temporal sequence are regularly interconnected. But there is an increasing number of applications where data are represented as graphs with complex relationships and non-Euclidean properties, for example in biology, chemistry, e-commerce, image understanding [33], social networks [19] and text analysis [34]. The complexity of graph data is challenging for machine learning approaches: a graph can be irregular, can have a variable size of nodes, nodes can have a different number of neighbors, so that easy operations in image domain can be difficult to be applied in graph domain. Several approaches to train models on graph structured data have been used, however those methods were used in pre-processing step to simplify the information, and they were not part of the training process. Graph Neural Network (GNN), a recent novel technique, is an end to end machine learning model that allows to learn a representation of graph structured data and to fit a predictive model on them.

GNNs [2] create an embedding of the graph nodes in a low dimensional space and the training process allows to learn the structural properties of the graph. This representation of nodes as an embedding can be created aggregating information from the neighbors of each node. When the task is classification or regression at node level, this embedding can be used directly. Otherwise, if task is to classify or predict values at graph or subgraph level, pooling techniques must be used to obtain a graph level representation.

Guo et al. in [33] proposed to use graph neural networks for image understanding so that features may interact and exchange information. For each image, there could be different cue types of interest for the understanding task, for example facial cues, body cues, object cues or whole image cues. For each cue type i, N_i features are extracted using deep models. Each feature represents a node and

every pair of nodes is connected by an undirected edge. They used a graph neural network (GNN) to update node messages by neighbors aggregation through a trainable nonlinear function depending on the hidden states of the neighbor nodes. At every timestep k, a gated recurrent unit (GRU) updates hidden states by taking as input the previous hidden state h_i^{k-1} and a message m_i^k. Finally, the last hidden states are pushed through a fully-connected (FC) layer followed by a Softmax layer to generate C class probabilities.

Ghosal et al. in [34] presented an emotion recognition in conversation (ERC) method based on graph convolutional neural network (GCN). This framework consists of three components: a Sequential Context Encoder to encode utterances of each speaker through a bidirectional gated recurrent unit (Bi-GRU), a Speaker-Level Context Encoder to learn dependencies in a conversation building a graph from encoded utterances, and an Emotion Classifier. First of all, a convolutional neural network (CNN) is used to extract textual features from the transcript. Secondly, since conversations are sequential by nature, these are the input for a Bi-GRU to capture contextual information that is speaker agnostic. Then, to capture speaker dependent information in a conversation they built a directed graph and used a R-GCN to transform the features. Finally, the contextually encoded and the speaker-level encoded features are concatenated and classified using a fully-connected network. For this purpose, they used a relational graph neural network similar to [35].

As already said, Liu et al. in [19] used a Pyramid Graph Convolution Network (PGCN) for their social relations analysis. The input for their graph based neural network was three graphs that model the appearance variance of the same person through the video, the interactions between different people and the interactions between a person and an object in the scene. They preferred a PGCN because a GCN can capture a global view in temporal domain, but an important

action could be overwhelmed by unimportant information. On the other hand, a PGCN can learn both long-term and short-term information performing relation reasoning with multi-scale temporal receptive fields.

To the best of our knowledge, this is the first work that classifies conversational type in egocentric social interactions based on a relational graph convolutional neural network (R-GCN) model.

Chapter 3

Proposed approach

Overview

The goal of the method developed in this Thesis is to take an egocentric video as input and classify the type of social interaction during a time interval. In this section, we present the features we extracted and the adopted model to solve this classification problem. For each person in the scene $[P_1, ..., P_M]$ we need two different types of features:

1. Non-relational features: for each person P_i in the scene, these features take into account information that belong to that specific person, independently of other interacting people. For our purposes, we need 3D head orientation and 2D head localization of each individual in the scene and the camera-wearer. The first-person is supposed to look at the space in front of him because of the head-mounted camera.

2. Relational features: for each person P_i in the scene, these features take into account individuals around him or her (including the camera wearer), to understand with whom he or she is exchanging social signals. These

relational cues are:

- Distance: based on distance from the camera estimated with a polynomial regression model by using face height.
- Attention: it is based on 3D head orientation and 2D head position to estimate the line of sight for each person.

The implemented model is based on Graph based Neural Network to model the relation between individuals and a Gated Recurrent Unit to capture the temporal changes.

3.1 Problem formulation

Given N video frames from a head-mounted camera as input, the output will be the class of social interaction for these frames. A graph for each frame is built by using not-relational features as node features and relational cues as edge features.

In particular, each graph represents a specific frame and a node V for every person in that interaction is present. In fact during an interaction a person may move out of the scene, but assuming his location does not change meaningfully, his features are interpolated between frames where he is present.

Each graph has two types of edges E: distance and attention. Distance edge value represents the distance between different individuals in the scene, while attention edge is a directed edge pointing from an individual to whom he is looking at. The output for each graph is a list of node embeddings. This list is vectorized and then concatenated with dynamic features at frame t in a vector g_t. The collection of vectors for each frame is the input of a GRU, which captures the sequential context information and the temporal changes. Finally, after a classification step we obtain the class for the social interaction in question.

3.2 Feature extraction

An intelligent feature extraction is the basis for a successfully automatic detection or classification problem. To completely understand a social interaction, a lot of different cues are needed, like the line-of-sight of people in the scene, pairwise distances, faces location and orientation. First of all, social interaction intervals must be identified, where each interval represents a single interaction. To extract all these different social cues for each interval, we proceed in different ways which will be analysed in the following subsections.

3.2.1 Microsoft Azure Cognitive Service

Microsoft Azure Cognitive Service is a very useful API[1] for face detection, person grouping, 3D head orientation and 2D head localization estimation. Microsoft provides a free limited key in order to develop projects. The Face–Detect API call provides a unique identifier string for each detected face in an image with a set of attributes. A lot of attributes can be detected, such as age, emotion, facial hair, gender, glasses, hair, head pose and so on. At a minimum, each detected face corresponds to a `faceRectangle`, that is a set of pixel coordinates to get the location of the face and its size. This service provides also four face recognition operation: for example, a Verify call takes two face IDs and determines if they belong to the same person, instead a Group call takes an array of face IDs and returns the same IDs grouped into several smaller arrays, where each group contains face IDs that appears similar. Nevertheless, Group API needs to be trained to perform a grouping operation, so it cannot be used because we have not a training set of faces available. To cluster faces found in the videos, I initialize groups with first faces found, and then I use the Verify API comparing

[1] https://azure.microsoft.com/en-us/services/cognitive-services/api

Figure 3.1: People grouping: in different frames, people can be grouped by comparing faces and a unique ID is assigned for each person

new faces with existing one to assign them a group. For each group I assign an ID (an increasing integer value starting from 1). In conclusion, by using the Microsoft Azure API, the following features are extracted:

- Groups of people with similar faces: each person has a unique ID to study the movements during a time interval.
- Face localization and size: height will be useful for distance estimation and the whole bounding box for the localization of the face in the image space.

```
top = face['faceRectangle']['top']
left = face['faceRectangle']['left']
width = face['faceRectangle']['width']
height = face['faceRectangle']['height']
```

- 3D Orientation: it returns Pitch – Roll – Yaw orientations in degrees. The value ranges are $[-90°, 90°]$, $[-180°, 180°]$ and $[-90°, 90°]$, respectively (as we can see in Figure 3.2).

```
pitch = face['faceAttributes']['headPose']['pitch']
yaw = face['faceAttributes']['headPose']['yaw']
roll = face['faceAttributes']['headPose']['roll']
```

Frames are elaborated with a 5 FPS rate instead of 15 FPS because of usage limits of Microsoft API. Assuming that people cannot move so fast, this decision does not lead to a loss of accuracy.

3.2.2 Distance from camera

After face detection, the distance of people in the video from the camera holder must be estimated. The main problem was that some parameters we needed were not written in the dataset's paper. So, we needed to do some experiments in order to estimate them. In particular, distance can be estimated using the height of detected face using this formula:

$$d = c/h$$

where c is a parameter obtained by camera calibration. In order to get that parameter, we used a GoPro camera on a tripod and we took photos of several

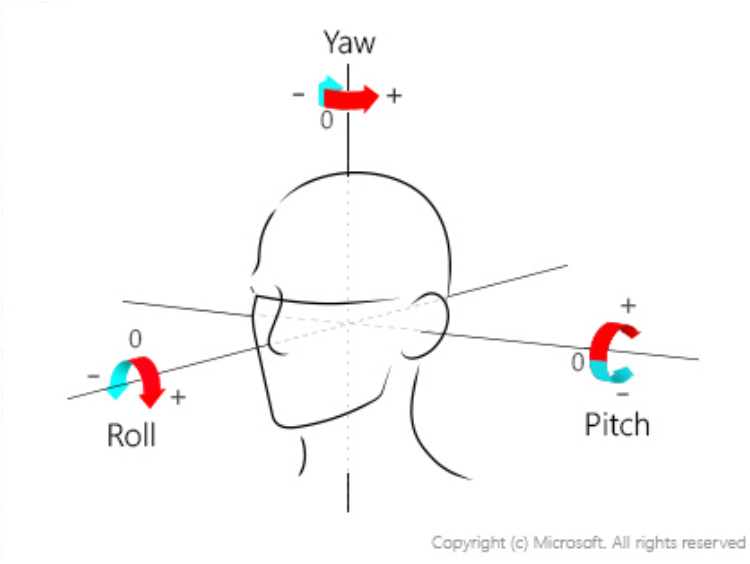

Figure 3.2: 3D head orientation

people at several distances (20–30–50–80–100–130–200 cm). In order to use these results, these photos had to be resized to the dataset images resolution. After that, a Polynomial Regression model was trained using face heights as input features and measured distances as labels. In this way, the slope of the obtained curve represents the c value we were looking for. The main problem in not having the real c parameter is that distances will not be the exact ones, but for this study we need only qualitative distances of people and we don't care about few centimeters errors. The trained polynomial is a second order polynomial (Figure 3.3) to model the trend of the data. To overcome problem due to the parabola trend for large x-coordinates, distance for faces higher than 250 are truncated into a minimum distance of 10 cm.

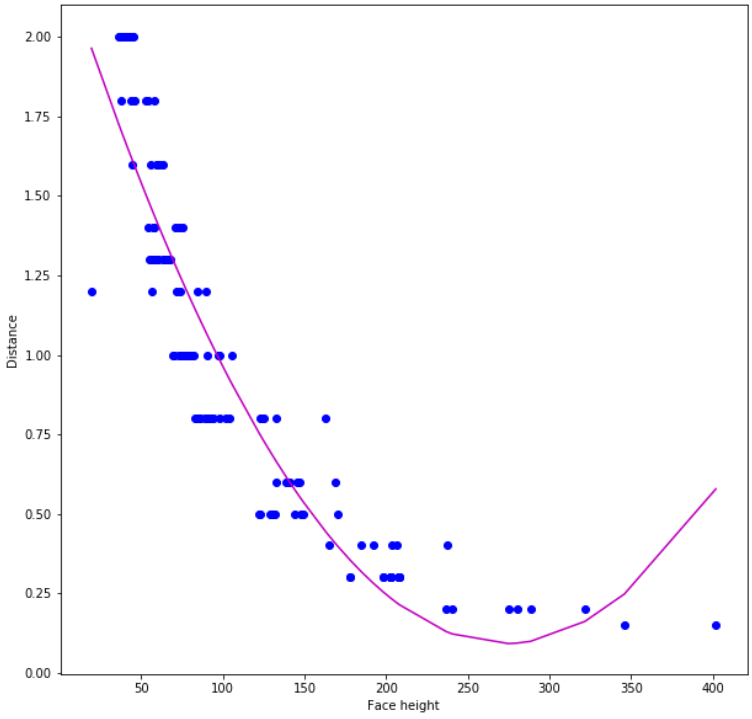

Figure 3.3: Polynomial regression for distance estimation

3.2.3 Pairwise Distance

Mutual distance estimation between people in the scene is very useful to understand what is happening and if there are some interactions. For this estimation the calculated distance from camera is very useful. In fact, if we represent people in the scene as colored circles in a x-d plane, where x is the x-coordinate in the estimated image plane width and d the distance from camera, Pythagorean theorem can be used to calculate mutual distance as hypotenuse of a right-angled triangle (Figure 3.4):

$$d = \sqrt{(x_2 - x_1)^2 + (d_2 - d_1)^2}$$

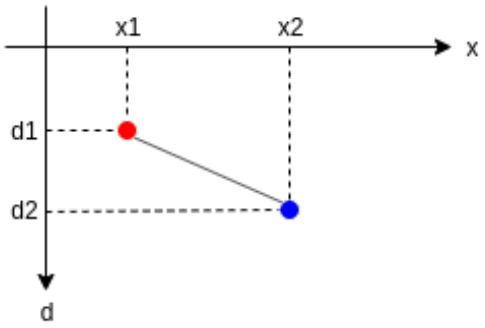

Figure 3.4: Pairwise distance estimation

3.2.4 Mutual Attention

An other important relational cue to understand a social interaction is the gaze direction and the mutual attention of people in a scene. In fact, a social interaction is characterized by pattern of attention between individuals. The main idea is that during an interaction, when someone speaks, he or she attracts attentions of others. Analysing these patterns of attention, different interactions can be individuated and classified. The first assumption for this task is to represent the gaze as a 3D cone in the space and looking for people inside this cone.

3.2.4.1 Notation

To describe the algorithm, we will use the following notation:

- V: it is the point in 3D space of the listener. The listener is who is looking at someone else, because we want to estimate to whom he is paying attention.
- R: it is the point in 3D space of the speaker in the scene who may attract attention of the listener.

3.2.4.2 Algorithm

Assuming that the gaze direction is coherent with the 3D head orientation, this feature is used to estimate mutual attention. People gazes can be described as a cone in 3D space with a certain angle, so that a person V is looking at someone else R if R's head point in 3D space belongs to the cone built from V's head orientation. In Figure 3.5 a 2D bird-view model is shown to represent this idea. If the cone condition is satisfied, a directed attention edge is tracked.
The algorithm steps are the following:

1. Define a unit vector u, which points along the axis of the cone (i.e. the first person line of sight based on his head orientation).
2. Define vector vr which points from the cone vertex V to the point in question R.
3. Take the dot product $vr \cdot u$. This value represents the cosine of the angle between vr and u.
4. Take the arc cosine of value calculated in 3. If the resulting angle is larger than the cone angle, the point is outside the cone.

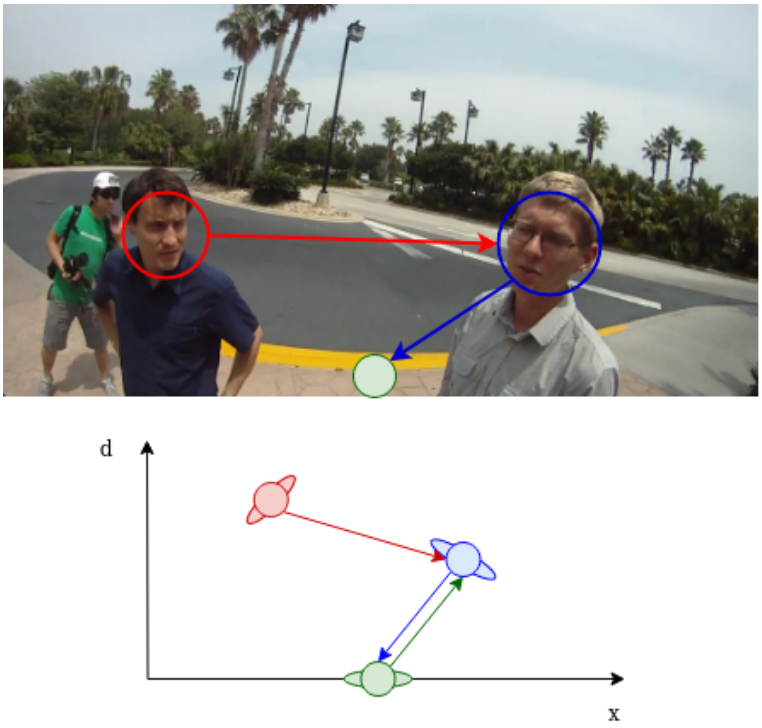

Figure 3.5: Bird-View Model for mutual attention definition

3.2.5 First-Person Head Motion

Besides faces, first-person motion is the other essential feature to classify social interactions. In fact movement patterns add context information and mainly allow to identify walking or not-walking interactions, since in the latter case faces are absent from most of the video. As already said, unpredictable movements of the camera wearer generates different motion patterns in the video, and these dynamic features can be captured by finding point correspondences between frames. Finally, a sequence of homographies can express these patterns between successive video frames.

3.2.5.1 Optical Flow

Optical flow is the pattern of apparent motion of objects in a visual scene as result of the relative motion between an observer and a scene. It can also be defined as the distribution of apparent velocities of movement of brightness pattern in an image. So, this method tries to estimate the motion between two ordered images which are taken at time t and $t + \Delta t$. If a point (x, y, t) with intensity $I(x, y, t)$ moves by Δx, Δy and Δt between two frames, a brightness constancy constraint can be defined as:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

Assuming small movements, this constraint can be developed as:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + H.O.T.^{1}$$

This equation cannot be solved due to the number of unknowns, so the introduction of additional conditions is needed for estimating the actual flow. There are a lot of different methods, we use Lucas-Kanade to find the point correspondences between frames. It assumes that the displacement of the images between two frames is small and approximately constant within a neighborhood of the point p under consideration. So, it solves the basic optical flow equations by using a window of pixels in that neighbourhood and computes the best motion approximation by the least squares criterion.

In summary, points at frame f_t are selected as in [36] and their correspondences are found at frame f_{t+1}. These correspondences are used as initial points to compute the next ones at frame f_{t+2}. Points can move out of the frames, so initial points could be computed again if the number of them is less than a threshold.

[1] High Order Terms from perturbation theory

Figure 3.6: Examples of motion features extracted from walking and not-walking interactions

In fact, next step is the estimation of the homography matrix, and the minimal number of points to compute is 4. For this project, this threshold is set to 6.

3.2.5.2 Homography estimation

A homography is a 3D transformation between 2 planes. So, a set of 2D point correspondences can be defined starting from two images showing the same scene with a different perspective. They can be determined by finding the coordinates in each image plane of a set of projected 3D points. Least squares method can be used to estimate the homography matrix between two images:

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H_{3\times 3} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

For each frame, we compute 15 homographies between 16 consecutive frames (the current one and the previous 15), which are normalized by the top-left corner element. This results in 15 3×3 homography matrices for each frame which are vectorized into a $m_t = \mathcal{R}^{135}$ vector. The stack of these normalized homographies is used to represent the global camera movement within an interval. In Figure 3.6 two examples of motion features are extracted: the image on the left shows a

walking interaction, while the other a not-walking one. These patterns are generated by drawing a line for each tracked point, connecting its positions through the frames. So, the difference between the images is clearly visible: the walking interaction leads to long lines representing the movement, while the not-walking one does not present clear pattern of movement but only few short lines.

3.3 Frame level modelling via R-GCN

3.3.1 Notation

Definition of a graph: A graph is defined as $G = (V, E, A)$, where V is the set of vertices or nodes, E is the set of edges and A is the adjacency matrix. Let $v_{\mathrm{i}} \in V$ denote a node and $e_{\mathrm{i,j}} = (v_{\mathrm{i}}, v_{\mathrm{j}}) \in E$ denote an edge pointing from v_{i} to v_{j}. The adjacency matrix A is a $n \times n$ matrix with $n = |V|$ and for each node it specifies connection (which can be weighted) with others:

$$\begin{cases} A_{\mathrm{i,j}} = w_{\mathrm{i,j}} = 1 & \text{if } e_{\mathrm{i,j}} \in E \\ A_{\mathrm{i,j}} = w_{\mathrm{i,j}} = 0 & \text{if } e_{\mathrm{i,j}} \notin E \end{cases}$$

The neighborhood of a node v is defined as $N(v) = \{u \in V | (u, v) \in E\}$. The degree of a node is the number of edges connected to it, formally defined as $degree(v_{\mathrm{i}}) = \sum A_{\mathrm{i},:}$. A graph may have node attributes X, where $\mathrm{X} \in \mathcal{R}^{n \times d}$ is a node feature matrix with $x_v \in \mathcal{R}^d$ representing the feature vector of node v. A graph may also have edge attributes $\mathrm{X^e}$, where $\mathrm{X^e} \in \mathcal{R}^{m \times c}$ is an edge feature matrix with $x^e_{u,v} \in R^c$ representing the feature vector of an edge (u, v).

Definition of a directed graph: A directed graph is a graph with all edges directed from one node to another. For a directed graph $A_{i,j} \neq A_{j,i}$, while for an undirected graph $A_{i,j} = A_{j,i}$. An undirected graph is a graph where all edges are

bidirectional.

3.3.2 Node embeddings

The first idea is to find embedding of nodes so that similar nodes in the graph are close in embedding space. So, the goal is to encode nodes and maps them to a low-dimensional vector so that similarity in this new space approximates similarity in the original graph. There are two key components:

- Encoder: it maps each node to a low-dimensional vector

$$\text{ENC}(v) = z_v$$

- Similarity function: it specifies how relationships in embedding space map to relationships in graph space

$$\text{similarity}(u, v) \approx z_v^T z_u$$

Different node similarity functions can be defined, based on how we decide if two nodes are similar, such as adjacency-based similarity (similarity function is the edge weight between u and v in the original network) or multi-hop similarity (which considers k-hop node neighbors). For the encoder, a simple approach is to use embedding-lookup:

$$\text{ENC}(v) = \text{Zv}$$

where

- $\text{Z} \in \mathcal{R}^{d \times |V|}$: embedding matrix, each column is node embedding
- $\text{v} \in \mathcal{I}^{|V|}$: indicator vector, all zeroes except in column indicating node v

Some limitations of this so-called "shallow" encoding exist, so other "deep" methods have been developed, which can be combined with all the existing similarity functions. Among all these deeper methods, we will discuss the basic one and the Graph Convolutional Networks (GCNs). In fact, for our project we use a special case of the latter one.

3.3.3 The basic Graph Neural Network

The first idea of GNN was presented by Gori et al. (2005) [37] and then elaborated by Scarselli et al. (2009) [38]. GNNs use the graph structure and node features to learn a representation vector of a node or the entire graph. Iteratively, the representation of a node is updated by aggregating representations of its neighbors. So, the key idea is to generate node embeddings based on local neighborhoods by using neural networks. In this way, every node defines a unique computation graph. After k iterations of aggregation, this representation captures the structural information within its k-hop neighborhood. Formally, the k-th layer of a GNN is defined as:

$$a_v^k = \text{AGGREGATE}^k\left(\{h_u^{k-1} : u \in N(v)\}\right)$$

$$h_v^k = \text{COMBINE}^k\left(h_v^{k-1}, a_v^k\right)$$

where h_v^k is the feature vector of node v at the k-th iteration and $N(v)$ is the set of nodes adjacent to v. The initialization step consists in setting $h_v^{(0)} = X_v$. The main difference is the approach to aggregate information across the layers, for example the basic approach uses average neighbor messages:

$$h_v^k = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}\right), \forall k > 0$$

where:

- $h_v^0 = x_v$: the initial embeddings are node features
- h_v^k: k-th layer embedding of v
- σ: non-linearity (e.g. ReLU or tanh)
- W_k, B_k: parameters matrices which are shared for all nodes
- $\sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|}$: average of neighbor's previous layer embeddings
- h_v^{k-1}: previous layer embedding of v

Two main properties of this approach can be outlined as:

1. Parameter sharing: aggregation parameters are shared among all nodes.
2. Inductive capability: if we train a network on a graph and a new graph has to be predicted, there is no need to retrain it. So it has generalization capacity for new graphs.

This iterative process is computationally intensive, so modern implementations try to speed it up, for example by limiting the number of iterations or by using random walks for processing the neighborhood.

As already mentioned, there are two types of tasks: node classification or regression and graph classification or regression. For the first task, each node $v \in V$ has a target value y_v and the goal is to learn a representation vector h_v of v so that its label can be predicted as $y_v = f(h_v)$. In this case, the node representation h_v^K of the final iteration is used for the prediction. For the second one, each entire graph G_i has an associated label y_i, the goal is to learn a graph representation h_G to predict the target value as $y_G = g(h_G)$. In this case, a READOUT function to aggregate node features from the final iteration is needed:

$$h_G = \text{READOUT}(\{h_v^K | v \in G\})$$

This READOUT function can be a simple summation or a more sophisticated pooling function, while the choice of the AGGREGATE and COMBINE functions is crucial.

3.3.4 Graph Convolutional Neural Network

Graph Convolutional Networks (GCNs) are methods to generalize the operation of convolution from grid data to graph data, i.e. from perfectly regular graphs (e.g. images) to irregular ones. They differ from graph embedding methods: instead of transforming a graph to a lower dimension, convolutional methods are performed on the entire input graph itself. Images can be seen as regular graphs, so the idea is to generalize locality and invariance for generic graphs. As illustrated in Figure 3.7, a graph convolution can be generalized from a 2D convolution. In 2D convolution, each pixel is seen as a node and its neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the node along with its neighbors. The graph convolution takes the weighted average value of the node features along with its neighbors, but different from image data, the neighbors of a node are unordered and size variable. In fact, as already said, graphs have an irregular structure, they are more abstract than images and variables like node degree, proximity and neighborhood structure provide

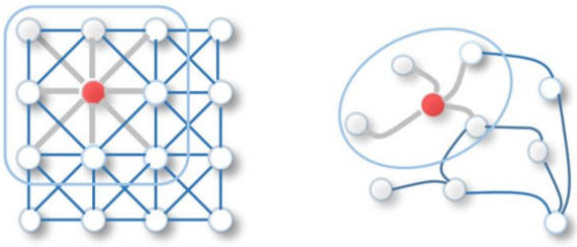

Figure 3.7: 2D convolution vs graph convolution [2]

.

information about the data. The main idea is to generate a node representation by aggregating its own features x_v and its neighbors' features x_u, where $u \in N(v)$. In Kipf et al.ś GCNs (2016) [39] the AGGREGATION function differs from basic GNNs as follows:

$$h_v^k = \sigma\left(W_k \sum_{u \in N(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

In particular we have:

- Same matrix W_k for self and neighbor embeddings is used
- Per-neighbor normalization instead of a single average

Instead, the previously mentioned COMBINE function can be expressed in matrix notation with the adjacency matrix and the weight matrix:

$$H^{k+1} = f\left(H^k, A\right) = \sigma\left(AH^kW^k\right)$$

where W^k is a weight matrix, A the already introduced adjacency matrix and $\sigma(\cdot)$ a non-linear activation function, like ReLU. This first approach is called spatial-based graph convolutional network, and it defines graph convolution based on node's spatial relations, by aggregating the neighbor nodes information to obtain a new representation for the node. In this approach, new representations are given by an iterative algorithm where some coefficients have to be updated with stochastic gradient descent. The first problem is that only neighbors' features are considered, while node features are not. In order to add its attributes, the adjacency matrix A have to be added to identity matrix I to create self-loops:

$$\hat{A} = A + I$$

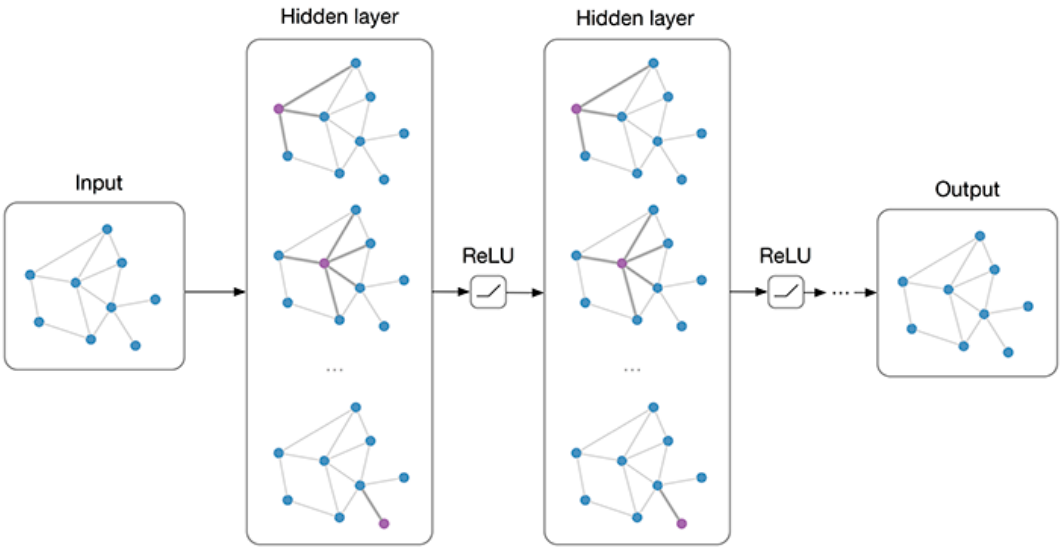

Figure 3.8: Graph Neural Network architecture from Kipf et al. 2016

.

Secondly, A is typically not normalized and therefore the multiplication with A will completely change the scale of the feature vectors. One way to normalize A is by taking the average of neighboring node features: $\hat{D}^{-1}\hat{A}$. In practice, a symmetric normalization is used: $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{\frac{1}{2}}$. In particular, $\hat{D}$ is the diagonal node degree matrix (i.e. number of neighbors per node). Finally, propagation rule can be defined as:

$$H^{k+1} = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{\frac{1}{2}}H^{k}W^{k}\right)$$

The second approach is spectral-based, and the first spectral-based GCN model was presented in Bruna et al. (2014) [40]. It is based on spectral graph theory, and now node features and attributes are represented by signals. Spectral-based approaches define graph convolutions by introducing filters from the perspective of graph signal processing, where the graph convolutional operation is interpreted as removing noises from graph signals. It is based on the eigen-decomposition of the Laplacian matrix, that is a mathematical representation of an undirected graph defined as:

$$L = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

where D is a diagonal matrix of node degrees. This Laplacian matrix is real symmetric positive semidefinite, so it can be factored as follows:

$$L = U \Lambda U^T$$

where $U = [u_0, u_1, ...] \in \mathcal{R}^{n \times n}$ is the matrix of eigenvectors ordered by eigenvalues and Λ is the diagonal matrix of eigenvalues, $\Lambda_{ii} = \lambda_i$. Directly solving a decomposition is intensitve, so many approaches approximate the spectral decomposition.

3.3.5 Relational Graph Convolutional Neural Network

Schlichtkrull et al. (2017) [35] introduced Relational Graph Convolutional Networks (R-GCNs) mainly to deal with highly multi-relational data. Using their same notation, directed and labeled graphs are denoted as $G = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, with nodes $v_i \in \mathcal{V}$ and labeled edges $(v_i, r, v_j) \in \mathcal{E}$, where $r \in \mathcal{R}$ is the relation type. This model is an extension of GCNs that operates on local graph neighborhoods. We use a propagation model similar to the one proposed by [35]. So, the model to update a node v_i in a relational, directed and labeled graph is an accumulated transformed feature vectors of neighboring nodes obtained as follows:

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

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i connected by a relation type $r \in \mathcal{R}$, a_{ij}^r is the edge weight and $c_{i,r}$ is a normalization constant for the specific edge. We choose this problem-specific normalization constant as $|N_i^r|$, that is the absolute value of the difference between number of ingoing and outgoing edges of node i belonging to a specific relation type r.

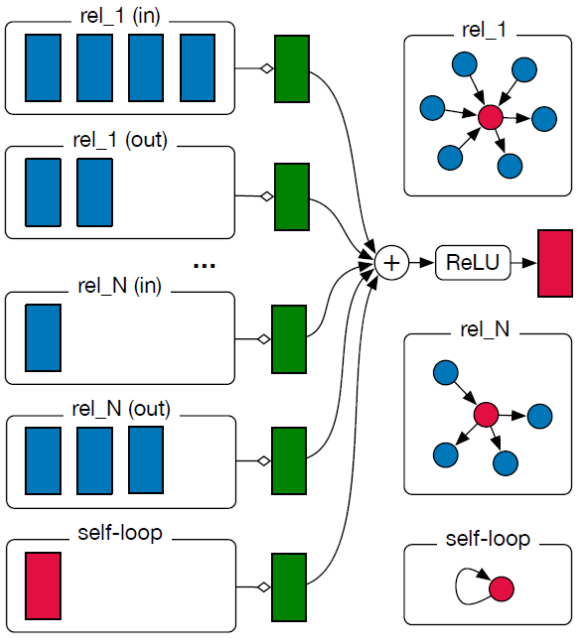

Figure 3.9: Diagram for nodes update in R-GCN model

The diagram for computing a single node update in R-GCN model is shown in Figure 3.9. Activations from neighboring nodes are gathered and transformed according to relation type and direction of the edge connecting to the node taken into account. All these transformed representations are summed and the sum passes through an activation function (such as ReLU). Parameters for this per-node update are shared across the whole graph.

Different from regular GCNs, as already said, this model introduces relation-specific transformations, depending on the type and direction of an edge. A special relation type is used to add self-loops for each node, so that a node representation at a specific layer can be propagated to its future layers. For this reason, it is suitable for our purpose because we want to define two different types of edges: distance and attention. By using this approach, for each edge we can define a direction, a type and a value, which will be used as normalization constant.

3.4 Temporal modelling via GRU

The temporal changes in the extracted features are crucial in understanding of social interactions. Tracking the evolution of face orientation or dynamical features may help in identify social patterns and differentiate classes of interaction. Recurrent Neural Networks are a generalization of feed-forward neural networks with an internal memory and they can use their internal state to process sequences of inputs. The standard feedforward neural network can only accept a single input of fixed dimensionality and map it to an output of fixed dimensionality. It cannot work with applications where the input is variable (e.g. frames in a video or text in one language) and/or the output is variable (e.g. text in another language). On the other hand, RNNs deal with sequential data by exploiting their internal memory: this makes them ideal for application such as speech recognition, time-series forecasting and conversation modeling. An other key is the parameter sharing across different timesteps, which allows the network to handle unseen input with different lengths and avoid overfitting across time. The relation between the current state of the network and the previous one can be written as follows:

$$a_t = g(a_{t-1}, x_t; W)$$

The current state a_t is a function of the current input x_t and the state of the previous timestep a_{t-1}, given a set of parameters W. This equation can be extended for multiple timesteps by keeping the same set of parameters W as follows:

$$a_{t+1} = g(x_t, g(x_{t-1}, ..., g(x_0, h_0; W); W); W)$$

As seen in Figure 3.10, a recurrent network takes the $x(0)$ from the sequence of input and it outputs $a(1)$. $x(1)$ and $a(1)$ are the input for the next step. In

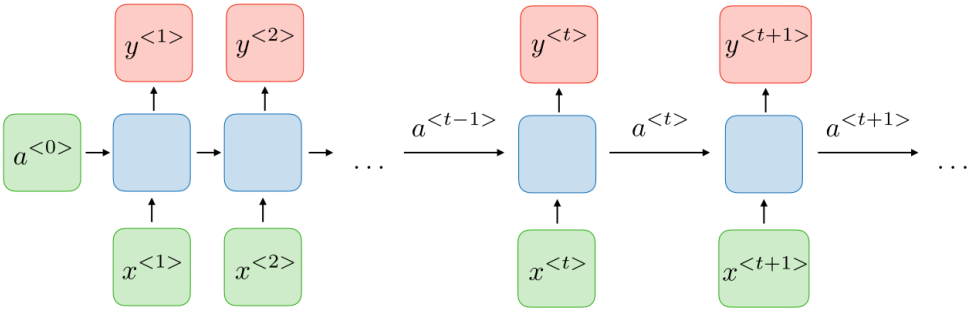

Figure 3.10: An unrolled recurrent neural network

this way, it keeps in memory the context while training. For each timestep t, the activation a_t and the output y_t can be calculated with the following formulas:

$$a_t = g_1(W_{aa}a_{t-1} + W_{ax}x_t + b_a)$$

$$y_t = g_2(W_{ya}a_t + b_y)$$

where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are the coefficients and g_1, g_2 the activation functions. A typical RNN architecture presents pros and cons, which can be summed up with the following list:

- Advantages:
  1. Weights are shared across time
  2. Processing input of any length
  3. Model size does not increase with input size
  4. Computation takes into account historical information
- Drawbacks:
  1. Slow computation
  2. Cannot consider any future input for current state

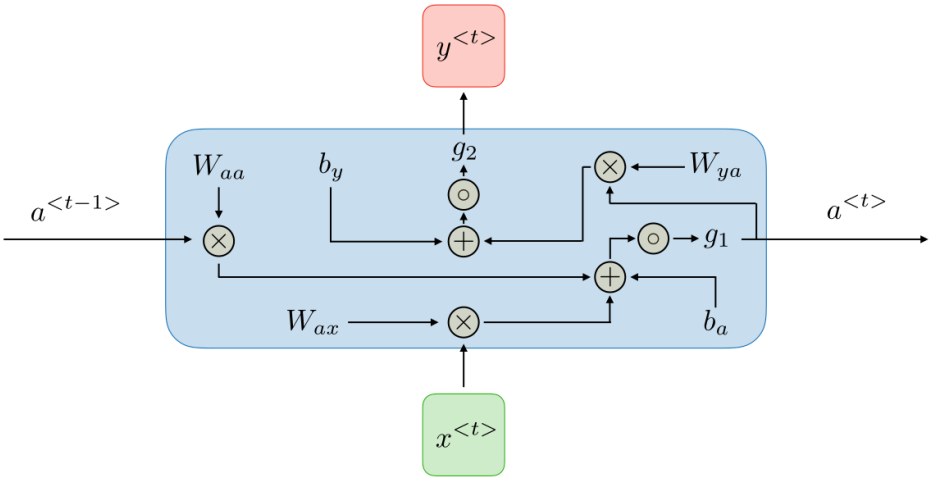

Figure 3.11: Description of a RNN block

3. Difficulty to capture long term dependencies (the so-called vanishing or exploding gradient phenomena)

Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) deal with the vanishing gradient problem of traditional RNNs.

3.4.1 Gated Recurrent Unit

For this project, GRU has been chosen because it has the same performances but higher efficiency and less complex structure in comparison to LSTM. GRUs were introduced by Cho et al [41] and they use the so-called update gate and reset gate, which allow to carry information over time, as shown in Figure 3.12 and explained below:

1. Update gate: this gate calculates z_t at timestep t with the formula

$$z_t = \sigma\left(\Gamma_{ux} \cdot x_t + \Gamma_{ua} \cdot a_{t-1}\right)$$

- The input x_t is multiplied by a weight Γ_{ux}

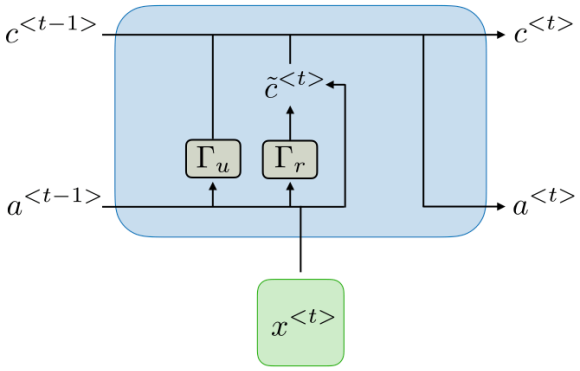

Figure 3.12: Gated Recurrent Unit

- The previous output a_{t-1}, which holds information from previous units, is multiplied by a weight Γ_{ua}
- Both are added together and a sigmoid function is applied to squeeze the output between 0 and 1

This gate eliminates the problem of vanishing gradient because the model on its own learns how much of the past information to pass along to the future.

2. Reset gate: this gate calculates the r_t at timestep t by using the following formula

$$r_t = \sigma\left(\Gamma_{rx} \cdot x_t + \Gamma_{ra} \cdot a_{t-1}\right)$$

- The input x_t is multiplied by a weight Γ_{rx}
- The previous output a_{t-1}, which holds information from previous units, is multiplied by a weight Γ_{ra}
- Both are added together and a sigmoid function is applied to squeeze the output between 0 and 1

This formula is very similar with the above gate and they differ only in the weights and the gate's usage. In fact, this gate has the opposite function-

ality: it is used by the model to decide how much past information has to be forgotten.

3. Current memory content: it uses the reset gate to store relevant information from the past as follows

$$\tilde{c}_t = \tanh\left(\Gamma_x \cdot x_t + r_t \odot (\Gamma_a \cdot a_{t-1})\right)$$

   - The input x_t is multiplied by a weight Γ_x and a_{t-1} by a weight Γ_a
   - Element-wise multiplication is applied to the reset gate r_t and the previous output a_{t-1}. This allows to pass only the relevant past information
   - Results from previous steps are added together and a tanh function is applied

4. Final memory at current time step: it has to compute the a_t vector which holds information for the current unit

$$a_t = (1 - z_t) \odot \tilde{c}_t + z_t \odot a_{t-1}$$

   - It applies element-wise multiplication to the update gate z_t and a_{t-1}
   - It applies element-wise multiplication to one minus the update gate $(1 - z_t)$ and current memory content $\tilde{c}_t$
   - Both are added together

So, GRUs are able to store or filter out information by using their update and reset gates and vanishing gradient problem disappears since the model is not washing out the new input every single time but keeps the relevant information and passes it down to the next time steps of the network.

3.5 Classification step

Before the classification step, a concat pooling similar to the one introduced in [42] has been chosen as pooling method. In simple terms, concat pooling means taking maximum and average value of the output of all timesteps, but unlike the method proposed by Howard et al. in [42], we do not concatenate along with the last hidden state because it would reduce the classification performances. Both Adaptive Max Pooling and Adaptive Avg Pooling are used due to the zero-padding, in fact the average value may be lower than the actual average and the maximum higher. A scheme of the pooling method can be seen in Figure 3.13.

For the classification task, a fully-connected layer is applied to reduce the dimensionality from the hidden size to the number of classes. Finally, a LogSoftmax layer gives probabilities for each class. Softmax Classifier is a generalization of Logistic Regression Classifier to multiple classes and gives normalized class probabilities, that is an intuitive output. So, a LogSoftmax can be expressed

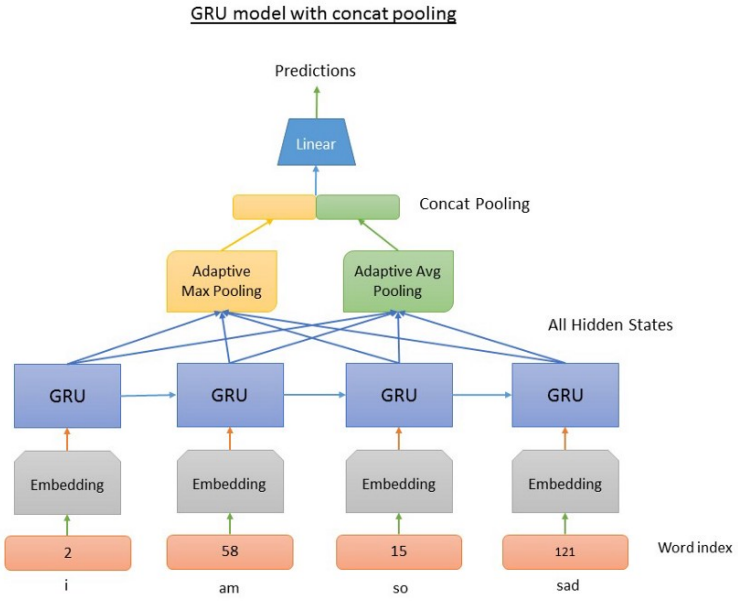

Figure 3.13: Example of concat pooling method

with the following formulation:

$$\text{LogSoftmax}(x_i) = \log\left(\frac{\exp(x_i)}{\sum_j \exp(x_j)}\right)$$

and it returns a tensor with the same shape of the input with values in range (-inf, 0). In conclusion, the class with the highest output is set as label of the video. Then, a negative log likelihood loss must be used for our purpose, because it expects an input containing log-probabilities of each class. It accepts also a weight tensor to assign weight to each of the classes, and this is particularly useful when you have an unbalanced training set.

3.6 The final model

In Figure 3.14 the workflow of the final model can be seen. For each interaction, each frame (timestep) is represented by a graph. Every node is a specific individual with a features list given by his non-relational cues, while edges are tracked by exploiting relational cues between interacting people in the scene. In particular, nodes are represented by a list of tensors (collected in a PyTorch stack) and their set of features consists of:

- 2D face localization (x–y coordinates): it can mainly help to identify monologues. In fact, during a monologue, individuals' positions can have a specific pattern around the speaker.
- 3D face orientation (yaw–pitch–roll coordinates): it helps to track social patterns and people attention.

For every edge, you must specify nodes index, a value, depending on distance and attention patterns, and a type, i.e. attention or distance. Distance edge is

a bidirectional edge because it describes the pairwise physical distance between two individuals, so for every node there must exist back and forth distance edges connecting to the others. On the other hand, an attention edge must be directed from a node to whom he is paying attention, so they may be not present if people in the scene are not looking at each other. Edge values must be expressed between 0 and 1 to provide a probabilistic interpretation of attention or distance levels. So that, higher the attention level and closer to 1 the edge value. On the other hand, nearer the people, closer to 1 the distance edge value. Edge indexes, values and types are collected in PyTorch stacks.

Once graphs are ready, each graph passes through a R-GCN which extracts a low-dimensional embedding at node-level to combine the description of nodes and their relationships. This embedding is now concatenated with the first-person head motion which compares point correspondences of the actual frames with the previous 15. This concatenation ensures to pay attention both to relational cues, static features and dynamical features. Only if we use all of these, we can maximize the average classification result, as we will se in the next section. A zero-padding is added to standardise the dimensionality to the maximum number of nodes, then this result becomes the input of a GRU to capture sequential context information. From the GRU output, maximum and average values are computed, so the pooling method explained in the last section can be used. Finally, a fully-connected layer reduces the dimensionality and a LogSoftmax outputs per-class log probabilities for the prediction.

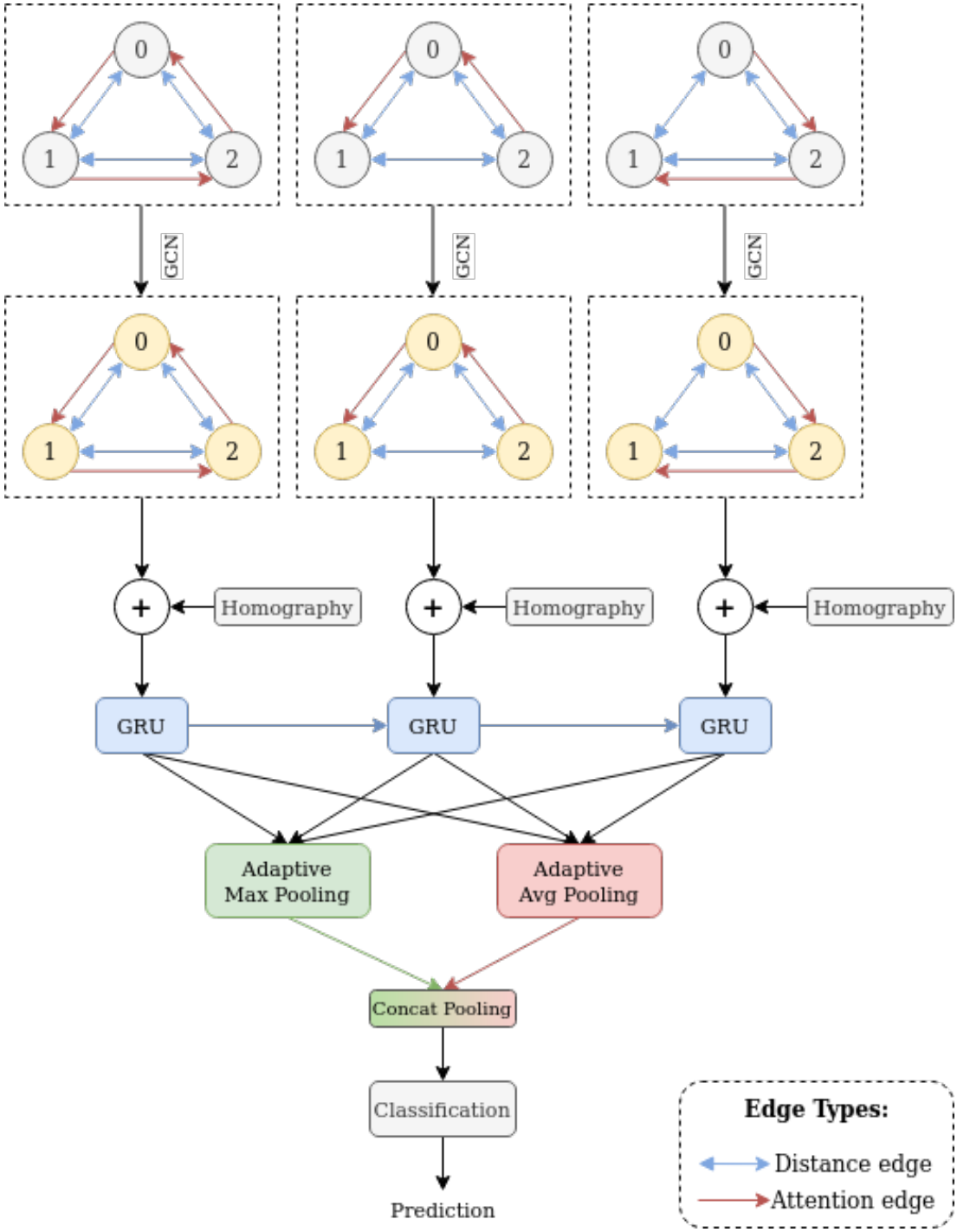

Figure 3.14: The final model used in this project

Chapter 4

Experimental Results

Summary

In this section we present our social interaction classification results, the implementation details, a visualization and a discussion of the results.

4.1 Dataset

As mentioned in Chapter 1, dataset had been recorded by Fathi et al. in [1] and it contains more than 42 hours of video recorded by 8 different subjects in Disney World Resort in Orlando, Florida. Each day a subset of the individuals used a head-mounted GoPro camera to record throughout the day, so each video contains a significant amount of experiences different from the other videos. The GoPro cameras were fixed on caps and they captured high definition 1280×720, 30 fps videos. Images have been extracted at 15 fps but features are extracted at 5 fps for API limit reasons. In addiction to videos, text files are provided describing the start and end time of intervals corresponding to a social interaction throughout the videos. There are five classes: dialogue, discussion, monologue, walk dia-

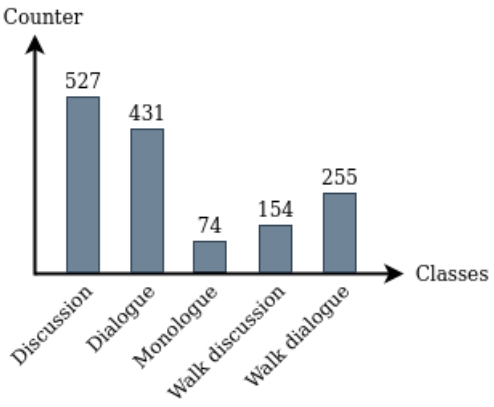

Figure 4.1: Histogram of label distribution

logue, walk discussion. Discussion means that multiple people are involved in the interaction and it is interactive, dialogue that only two people are involved and monologue that it is one-sided, i.e. everybody pays attention to a single speaker. Walk interactions occur when people interact while walking. Each of these interactions can take place at a dinner table with group of friends, while walking, or while standing in a line, etc. In total there are 1141 annotated interactions, whose label distribution is shown in Figure 4.1. As we can see, this dataset is highly unbalanced because we have big differences on the label distributions, so we have to handle this problem with methods as weight balancing and also using F1-score in addition to accuracy as metric for classification performance.

Figure 4.2 illustrates the difficulty of using this dataset to accomplish our task: different conversation types may occur at different time intervals, even if contextual information does not change, e.g. discussion vs dialogue. Secondly, since this dataset has been acquired in a naturalistic setting, interaction cues present a large intra-class variability as well as unpredictable camera movement. For example, in discussions or walk discussions there may be few or many people. During a monologue, a lot of people can be present and the speaker might or might not look at the camera wearer (so that the face may not be detected).

Figure 4.2: Example of video clips belonging to different conversation types in the dataset

Or an occlusion may occur due to other people, so the speaker can be unseen. So, if the speaker's face is not present, the monologue recognition becomes very challenging because it mainly relies on the unidirectional attention from people to the speaker. The day-long videos show a big variability in the scenes. For example, during a dialogue (i.e. an interaction only with one person), many people can be present because this interaction may occur in a queue or at a table. This variability can be represented also by the different light conditions, and if the video is recorded outdoor or indoor. Different conditions can lead to more difficulties in face detection.

4.2 Implementation details

4.2.1 Pre-processing

For each social interaction time interval, features are extracted as in Section 3.2 and three files are saved, where `videoName` is the name of the video in the dataset and `interaction` is the number of the interaction which indicates an interval of frames:

- `f_videoName_interaction`: it contains arrays of arrays. In particular, there is an array for each frame and an array for each person in that frame.

This last array contains all the person's features, such as his ID, head orientation, localization and distance from camera.

- `d_videoName_interaction`: it contains arrays of arrays. In particular, there is an array for each frame and an array for each pair of people in the same scene. This array contains two IDs and distance between them.

- `h_videoName_interaction.csv`: it contains 15 homography matrices for each frame.

Once features are extracted as in Section 3.2 and saved as just mentioned, a pre-processing is needed. In fact, we can make assumptions to use only meaningful features for the classification task. For example, in a single frame a person can disappear due to camera movements, but we assume that interacting people don't move significantly. So we calculate a presence level for each person given by the percentage of frames in which he appears weighted by the average distance from camera wearer, because nearest people are more likely to disappear:

$$presence\_level = \frac{percentage}{avg(d)}$$

We use this value to filter out individuals with a low level of presence in the video because probably they are not interacting. In training set, for dialogue and walk dialogue interactions we keep only the most present person in the video. For the others, we find the ID with the maximum level max and keep people with a level higher than 20% of max. Then, we interpolate all the individuals' features in temporal adjacent frames for the assumption of static position of people during an interaction. Mutual attention is now calculated by using localization and face orientation, as seen in 3.2.4 by using algorithm in 3.2.4.2.

For its implementation, the first thing to do is to convert Euler angles to Cartesian coordinate system using Python `math` module:

```
u_x = math.cos(yaw) * math.cos(pitch)
u_y = math.sin(yaw) * math.cos(pitch)
u_z = math.sin(pitch)
```

Roll coordinate is not used because it does not affect directional vector. After all the algorithm steps, the attention level is expressed with a value between 0 and 1 in a probabilistic view according to true angle between the first person's line of sight and the second person's position. The angle for the cone (*aov*, angle-of-view) is chosen by considering the near-peripheral vision[1], that is 60° (30° if we consider the angle starting from the center of gaze). If the point is outside the cone (i.e. calculated angle *theta* is greater than $\frac{aov}{2} = 30°$), the attention level is set to 0. Otherwise, the probability is calculated with $1 - \frac{theta}{aov/2}$, so that more *theta* is close to 0°, more the attention level is close to 1. For each person, we always keep the maximum level and all the others if higher than 0.5. Since also the distance edge values must be a value between 0 and 1, it is taken as $1 - \frac{d}{max_i(d_i)}$ that is the pairwise distance divided by the highest distance.

Finally, features are interpolated to have the original frame rate (15 fps), so that they can be compared to the homographies dimensionality. Once pre-processing is done, training set is divided in 200-frames intervals to standardize the training process. According to the length of each interaction, a specific number of intervals is kept and only features belonging to these frames are collected.

Dataset is now split in training, validation and test sets. First of all, to evaluate overfitting we keep 80% as training set and the remaining 20% as validation set. Finally, about 1700 intervals belonging to 5 subjects videos are used for training and the remaining 3 subjects for testing, as in [1].

[1] `https://en.wikipedia.org/wiki/Peripheral_vision`

4.2.2 Feature scaling

A feature scaling approach is very useful when feature values range varies a lot among different features. The result of a standardization is that rescaled features have $\mu = 0$ and $\sigma = 1$, where μ is the mean and σ the standard deviation. Values needed for this operation are calculated by gathering only features of training set. Finally, features from both training and test set are rescaled with the following formula:

$$z = \frac{x - \mu}{\sigma}$$

4.2.3 Data augmentation

To obtain better results in our classification task, large amount of training data is needed, so we apply the data augmentation approach proposed by [31] directly on feature vectors. Both nodes and edges' features are flattened in a single vector and the main idea is to add slight variations to the data by computing PCA and adding random noise in the direction of the eigenvectors and proportional to the eigenvalues of the feature matrix, multiplied by a Gaussian random variable $X \sim \mathcal{N}(\mu = 0, \sigma = 0.001)$. Thanks to this approach, we can double the number of training intervals, so we are able to obtain additional samples and preserve original labels.

4.2.4 Hyperparameters tuning

A grid search is implemented to get the best parameters, such as learning rate, hidden size, graph nodes output size and weight decay. A single validation set is used, the model is trained for 10 epochs and the minimum value of average loss is used to choose the parameters. At the end of tuning process, we have selected the following parameters:

- learning rate = 10^{-6}
- hidden size = 64
- graph nodes output size = 2
- weight decay = 0.005

In particular, different values of learning rate and weight decay brought to a high loss of performance.

4.2.5 Graph construction

For the graph construction, we need to stack the features of all the nodes. Then, for the R-GCN we need 3 different tensors to define edges in the graph:

- `edge_index`: it contains a list of source node indices and an other of target node indices and it is used to define the direction of the edges.
- `edge_type`: it contains a list of types for each edge. In this case, type 0 is related to distance edges and type 1 to attention edges.
- `edge_norm`: it contains a list of edge's attributes. Every edge is associated with a value that may be distance or attention level. Each value is between 0 and 1, and it is normalized by using the absolute value of the difference between the number of ingoing and outgoing edges.

4.2.6 Experimental settings

The optimization algorithm chosen to update the network weights is the so-called AdamW. The original version, Adam, was presented by Kingma et al. [43]. It is an stochastic gradient descent algorithm based on adaptive estimation of 1st and

2nd-order moments, it is computationally efficient and has little memory requirements. AdamW is an evolution of the original one proposed by Loshchilov et al. in [44]. In their article, they demonstrate that L2 regularization is not equivalent to weight decay and it is significantly less effective for adaptive algorithms than for SGD. So the improved version of Adam should generalize better and should be able to compete with SGD. In order to reduce overfitting, we add a L1 regularization with the L1Loss criterion by `torch.nn` module. Then, to handle imbalanced dataset problem we use weight balancing, that balances data by providing higher weights to less frequent classes. The most frequent class (i.e. discussion in our case) has a weight equal to 1, while the other ones have weights greater than 1 depending on their frequency. This weight tensor is used as optional argument by the loss function to give more or less importance and to perform better. For the classification, a softmax and a logarithm are then applied: for computational reasons, `log_softmax` function from `torch.nn.functional` is used because it is mathematically equivalent to $\log(\text{softmax}(x))$, but doing these two operations separately is slower and numerically unstable. This function uses an alternative formulation to compute the output and gradient correctly. So, the loss function is a `nll_loss` from `torch.nn` module, that is a negative log likelihood loss, useful to train a classification problem with C classes. Its input is expected to contain log-probabilities of each class, that is the output of the LogSoftmax layer, while the target is a class index in the range $[0, C-1]$ where C is the number of classes. Regarding the metrics, we use both accuracy and F1-score because of the unbalanced dataset and the best result is achieved after 83 epochs.

Model	Accuracy	F1-score
MLP	44.03%	27.27%
HCRF [1]	38.99%	38.99%
R-GCN	46.98%	47.07%
GRU	53.02%	53.63%
InteractionGCN	**61.88%**	**62.36%**

Table 4.1: Performance comparison with the SOTA and other baselines.

4.3 Results

As seen in Table 4.1, various baseline methods are used to compare our result, and all of them take as input the features computed in Section 3.2. A first baseline is a Multilayer Perceptron (MLP) because it has proved to perform better than linear models for video classification purpose [45]. MLP model input is the concatenation of 3D head orientation, histogram of roles computed as in [1], first-persion motion features and its output is the class prediction for each frame. For the test set prediction, the most predicted class is chosen for the video clip. To validate the different components of our framework, we consider separately a GRU as temporal model and then a R-GCN to represent the social network structure of the interactions. For the GRU model we stack non-relational features and an histogram of roles at frame-level with first-person motion features and it obtains 53.02% accuracy and 53.63% f1-score. For the R-GCN model we stack the output of the graph neural network with the first-person motion features and it results in 48.56% accuracy and 47.56% f1-score. Regarding the SOTA [1], we do not have the final score from the original paper, so by using first-person motion, 3D face localization, histogram of roles for each frame we can recompute the results by feeding these features to a Hidden Markov Random Field (HMRF) model[1] with 10 hidden states as in [1]. The results we got are poorer

[1]https://github.com/yalesong/hCRF-light

than theirs by considering the reported confusion matrix. Probably, 38.99% of both accuracy and f1-score are due to the fact that our features are computed differently. Neverthless, our model achieves an average improvement of about 5% over the per-class recognition recall reported in [1]. Moreover, our experiments show that both R-GCN and GRU are able to improve classification performance and our overall framework achieves 61.88% accuracy and 62.36% f1-score.

4.4 Ablation study

In Table 4.2 there is a list about the effect of each feature employed in our model while in Table 4.3 there are the confusion matrices obtained by removing a feature at time. The largest performance drop is present by removing first-person motion features, where there is more confusion between dynamic and static interactions. In fact they are fundamental to distinguish walk interactions from static ones as seen with patterns drawn in Section 3.2.5 and as we can see in the related confusion matrix. Removing head localization or orientation seem to bring to a slightly higher score, but analysing the corresponding confusion matrices, removing one of those features bring to an important loss in monologues classification. Also removing the others brings to a loss in monologue classification. So this type of interaction results to be the most difficult in a classification problem, due to

Removed cue	Accuracy	F1-score
2D head localization	62.47%	62.45%
3D head orientation	62.47%	62.67%
Mutual attention	59.84%	60.34%
Pairwise distance	62.20%	62.53%
First-person motion	49.71%	48.89%
All cues	**61.88%**	**62.36%**

Table 4.2: Performance comparison by removing a feature at time.

Removed cue	Discus.	Dial.	W. Disc.	W. Dial.	Monol.	Avg score
H. localization	62.5%	68.6%	58.6%	63.1%	17.6%	54.08%
H. orientation	62.5%	68.6%	55.2%	66.2%	23.5%	55.20%
M. attention	61.3%	60.8%	48.3%	66.2%	35.3%	54.38%
P. distance	63.1%	70.6%	65.5%	55.4%	23.5%	55.62%
F. p. motion	57.7%	52.0%	10.3%	40.0%	41.2%	40.24%
All cues	**57.7**%	**68.6**%	**62.1**%	**66.2**%	**41.2**%	**59.16**%

Table 4.3: Per class classification recall comparison by removing a feature at time.

the fact that it relies on complex patterns such as the attention of the individuals towards who is talking and on their physical position around the speaker. Overall, all the features together improve from 4% to 19% the average per-class classification score.

4.5 Discussion

Experimental results show that the recognition of conversational type from video clips is a challenging task, most of all when it is focused only on detected faces in the frames. Figure 4.3 shows the confusion matrix obtained with our model and it can be compared to confusion matrix reported in [1]: our recall results are better in general, except for monologue, and the average per-class score improves of about 5%. Firstly, this framework allows a well discrimination of walk interactions and static ones thanks to the dual use of first-person motion features and GRU. Main confusion among classes is present between discussion–dialogue and between walk discussion–walk dialogue, because there is a common contextual information that may bring to error. On the other hand, the most misclassified class is monologue because it relies on complex structures and patterns of interacting individuals which are not easy to find. As already said, monologue is one-sided interaction where a speaker is talking while others are listening. In a realistic

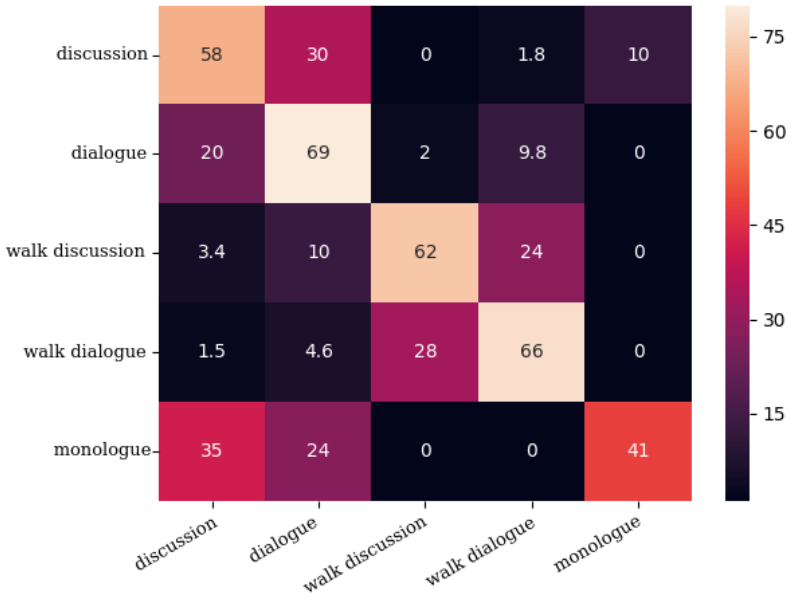

Figure 4.3: Confusion matrix obtained with our model

scenario, many people may bring to higher occlusion or faces may be invisible from the camera viewpoint. It could be a big problem especially if the speaker is not visible. The loss of performance for this class is likely due to the fact that our approach on identify mutual attention is less sophisticated than the one used in [1], because it strongly relies on detected faces and does not estimate the presence of invisible individuals. Fathi's model is able to identify when people are looking at a common location in the space, even in absence of a detected face in such location, so it is more robust to occlusions. In addition, this is the class with the smallest number of examples in the dataset, as seen in Figure 4.1.

In conclusion, because of the unconstrained nature of the videos, face detection may fail, for example when people wear accessories (e.g. caps or glasses), when an occlusion does not let the camera see the speaker or simply when the speaker looks the other way and the camera can see only his/her back and not his/her face.

We believe that improving this model by adding other input cues could strongly help the discrimination of ambiguous classes. Some of these cues can be the pairwise body poses, including the pose estimation of the invisible camera wearer as proposed in [46].

Chapter 5

Conclusions

In this work we presented a graph-based neural network architecture to expand the state of the art in social interaction classification in egocentric videos. Our model aims to classify a video clip in a conversational type such as discussion, dialogue, walk discussion, walk dialogue or monologue. Firstly N frames are collected, then features are extracted at 5 FPS for API usage limits. Faces are the main source of information, in fact features are extracted for each detected face in that video frame. They can be categorized into non-relational or relational features. The non-relational cues take into account information about each person independently to others, and we used 3D head orientation and 2D head localization. They may help in understanding patterns of position and attention of people. On the other hand, relational cues model pairwise relations between people in the scene. We used pairwise distance and mutual attention, describing each gaze as a 3D cone in the space. All these features must be extracted for every interacting individual in the scene but also estimated for the camera wearer, assuming that he is looking at in front of him due to the head-mounted camera. For each frame, features are used to build a graph representing the social network structure in the video, where a person corresponds to a node, non-relational cues are the node features and relational cues are used to track the edges. For each

edge we have to specify relational type, direction and value. In this case, we have two relational types, i.e. attention or distance. Attention edges are directed, while distance edges are bidirectional. Edge values must be between 0 and 1, depending on the importance of that connection. For this reason, higher the attention level or lower the pairwise distance, higher the value of that edge. The framework is able to capture the social network structure by using a R-GCN, which extracts nodes embedding depending on both features and relationships. At the same time, homography matrices are computed by looking for point correspondences between the current frame and the previous 15 frames. Then, the concatenation of the R-GCN output and the first-person motion features feeds a GRU to extract sequential context information. This allows to take into account evolution of people features and dynamic motion features and changes of patterns of position and attention,. We validated this framework with the first-person dataset gathered by Fathi et al. in [1] and we compared our results with various baselines. Experimental results present an important improvement compared to the SOTA, Multi-Layer Perceptron, R-GCN-only and GRU-only. In particular, there is an improvement of about 5% over the per-class recognition recall compared to the SOTA [1] and an improvement of 35.09%, 15.29% and 8.73% over the F1-score compared to the different baselines.

The largest limits we analysed are related to the simple features we used as input and some methods for relational cues estimation. For example, mutual attention estimation may be improved by considering not only detected faces but also 3D body poses. This would help when people faces are not visible, so that attention estimation fails. We believe that having a more robust attention estimation may help in understanding of different conversational types.

Future work will be focused on improving the quality of input features by exploiting also 3D body pose estimation as source of information. Body pose will

be estimated not only for visible people in the scene, but also for the invisible camera wearer as proposed in [46]. Finally, a new method for relational cues estimation may be explored, for example the method proposed in [1].

References

[1] A. Fathi, J. K. Hodgins, and J. M. Rehg, "Social interactions: A first-person perspective," in *Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2012. ii, 7, 8, 15, 48, 52, 56, 57, 58, 59, 62, 63

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019. 3, 16, 34

[3] M. Pantic and A. Vinciarelli, "Social signal processing," in *Oxford handbook of affective computing*, p. 84, 2014. 6

[4] M. Cristani, R. Raghavendra, A. Del Bue, and V. Murino, "Human behavior analysis in video surveillance: A social signal processing perspective," in *Neurocomputing*, pp. 86–97, 2013. 6, 11

[5] Y. Chen, W. H. Hsu, and H. Y. M. Liao, "Discovering informative social subgraphs and predicting pairwise relationships from group photos," in *Proceedings of the 20th ACM international conference on Multimedia*, pp. 669–678, ACM, 2012. 10

[6] Q. Dai, P. Carr, L. Sigal, and D. Hoiem, "Family member identification from photo collections," in *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pp. 982–989, 2015. 10

[7] A. Dehghan, E. Ortiz, R. Villegas, and M. Shah, "Who do i look like? determining parent-offspring resemblance via gated autoencoders," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1757–1764, IEEE, 2014. 10

[8] Y. Guo, H. Dibeklioglu, and L. van der Maaten, "Graph-based kinship recognition," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 4287–4292, IEEE, 2014. 10

[9] M. Shao, S. Xia, and Y. Fu, "Identity and kinship relations in group pictures," in *Human-Centered Social Media Analytics*, pp. 175–190, Springer, 2014. 10

[10] P. Singla, H. Kautz, J. Luo, and A. Gallagher, "Discovery of social relationships in consumer photo collections using markov logic," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW 08. IEEE Computer Society Conference on*, pp. 1–7, IEEE, 2008. 10

[11] G. Wang, A. Gallagher, J. Luo, and D. Forsyth, "Seeing people in social context: Recognizing people and social relationships," in *European conference on computer vision*, pp. 169–182, Springer, 2010. 10

[12] S. Xia, M. Shao, J. Luo, and Y. Fu, "Understanding kin relationships in a photo," in *IEEE Transactions on Multimedia, 14(4)*, pp. 1046–1056, IEEE, 2012. 10

[13] T. Shu, D. Xie, B. Rothrock, S. Todorovic, and S. C. Zhu, "Joint inference of groups, events and human roles in aerial videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages*, p. 45764584, IEEE, 2015. 11

[14] T. Lan, L. Sigal, and G. Mori, "Social roles in hierarchical models for human activity recognition," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1354–1361, IEEE, 2012. 11

[15] V. Ramanathan, B. Yao, and L. Fei-Fei, "Social role discovery in human events," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2475–2482, IEEE, 2013. 11

[16] J. Zhang, W. Hu, B. Yao, Y. Wang, and S. C. Zhu, "Inferring social roles in long timespan video sequence," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 1456–1463, IEEE, 2011. 11

[17] Q. Sun, B. Schiele, and M. Fritz, "A domain based approach to social relation recognition," in *International conference on Computer Vision and Pattern Recognition(CVPR)*, 2017. 11, 12

[18] D. B. Bugental, "Acquisition of the algorithms of social life: A domain-based approach," in *Psychological Bulletin, 126(2)*, p. 187, 2000. 11

[19] X. Liu, W. Liu, M. Zhang, J. Chen, L. Gao, C. Yan, and T. Mei, "Social relation recognition from videos via multi-scale spatial-temporal reasoning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3566–3574, IEEE, 2019. 12, 16, 17

[20] Y. J. Lee, J. Ghosh, and K. Grauman, "Discovering important people and objects for egocentric video summarization," in *IEEE Conference on Computer Vision and Pattern Recognition*, p. 13461353, IEEE, 2012. 12

[21] D. Damen, T. Leelasawassuk, and M.-C. W, "You-do, i-learn: Egocentric unsupervised discovery of objects and their modes of interaction towards

video-based guidance," in *Computer Vision and Image Understanding, vol. 149*, p. 98112, 2016. 12

[22] M. Ryoo and L. Matthies, "First-person activity recognition: Feature, temporal structure, and prediction," in *International Journal of Computer Vision, vol. 119, no. 3*, pp. 307–328, 2016. 12

[23] Y. Yan, E. Ricci, G. Liu, and N. Sebe, "Egocentric daily activity recognition via multitask clustering," in *IEEE Transactions on Image Processing, vol. 24, no. 10*, p. 29842995, IEEE, 2015. 12

[24] S. Alletto, G. Serra, S. Calderara, and R. Cucchiara, "Understanding social relationships in egocentric vision," in *Pattern Recognition, 48(12)*, pp. 4082–4096, 2015. 12, 13

[25] S. Narayan, M. S. Kankanhalli, and K. R. Ramakrishnan, "Action and interaction recognition in first-person videos," in *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–518, 2014. 12

[26] J. A. Yang, C. H. Lee, S. W. Yang, V. S. Somayazulu, Y. K. Chen, and S. Y. Chien, "Wearable social camera: Egocentric video summarization for social interaction," in *Multimedia Expo Workshops, International Conference*, pp. 1–6, 2016. 12

[27] H. Soo Park and J. Shi, "Social saliency prediction," pp. 4777–4785, 2015. 12

[28] H. S. Park, E. Jain, and S. Y, "3d social saliency from head-mounted cameras," in *Advances in Neural Information Processing Systems*, pp. 431–439, 2012. 12

[29] E. S. Aimar, P. Radeva, and M. Dimiccoli, "Social relation recognition in egocentric photostreams," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 3227–3231, IEEE, 2019. 12

[30] M. Aghaei, M. Dimiccoli, and P. Radeva, "With whom do i interact? detecting social interactions in egocentric photo-streams," in *Pattern Recognition, 23rd International Conference on*, p. 29592964, 2016, 2016. 13

[31] M. Aghaei, M. Dimiccoli, C. CantonFerrer, and P. Radeva, "Towards social pattern characterization in egocentric photo-streams," in *Computer Vision and Image Understanding, vol 171*, pp. 104–117, 2018. 14, 53

[32] S. Bano, T. Suveges, J. Zhang, and M. S. J, "Multimodal egocentric analysis of focused interactions," 2018. 14

[33] X. Guo, L. F. Polania, B. Zhu, C. Boncelet, and K. E. Barner, "Graph neural networks for image understanding based on multiple cues: Group emotion recognition and event recognition as use cases," in *2019 22nd International Conference on*, pp. 4287–4292, IEEE, 2014. 16

[34] D. Ghosal, N. Majumder, S. Poria, N. Chhaya, and A. Gelbukh, "Dialoguegcn: A graph convolutional neural network for emotion recognition in conversation," 2019. 16, 17

[35] M. Schlichtkrull, T. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," 2017. 17, 37

[36] J. Shi and C. Tomasi, "Good features to track," in *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, p. 600, 2000. 28

[37] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, p. 729734, 2005. 32

[38] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," in *IEEE Transactions on Neural Networks*, p. 6180, IEEE, 2009. 32

[39] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2016. 35

[40] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2014. 36

[41] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoderdecoder for statistical machine translation," 2014. 41

[42] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018. 44

[43] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," ICLR, 2015. 54

[44] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," ICLR, 2019. 55

[45] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014. 56

[46] E. Ng, D. Xiang, and H. Joo, "You2me: Inferring body pose in egocentric video via first and second person interactions," in *arXiv*, vol. abs/1904.09882, 2019. 60, 63

[47] S. Escalera, X. Baró, H. J. Escalante, and I. Guyon, "Chalearn looking at people: A review of events and resources," in *IEEE International Joint Conference on Neural Networks*, pp. 15794–1601, IEEE, 2017.

Acknowledgements

I would like to express my gratitude to Dr. Mariella Dimiccoli, for having followed and supported me during this work and for her constant help to solve every problem or doubt.

A special thanks to people in IRII for their kindness, especially to Perception and Manipulation Group who enriched this experience.

I want to express my heartfelt thanks to Naiara, for never having hesitated in helping me while I was far from home.

I would like to thank also Dr. Gabriele Costante for his support and for having introduced me to this investigation area.

Thanks to all my family for their unconditional support on my choices. To my parents, who never asked me anything but realize whatever I want, because the education, the moral principles, the "if you can, I can" helped me in growing up and being who I am today.
To Alessio, to my grandmothers, to my aunt, uncle and Sara for having shared with me both joys and worries during these years. To my grandfathers, I feel they would have been happy and proud.

To my friends, because University years would have been highly different without you.

Last but not least, to Marta, because you have always been there, both in happy and difficult times. With you, I am not afraid of the changes and challanges of life.

Ringraziamenti

Un ringraziamento speciale alla Dr.ssa Mariella Dimiccoli, che mi ha seguito e supportato durante tutto il lavoro, per il suo costante aiuto nella risoluzione di dubbi e problemi.

Grazie anche alle persone dell'IRII per la loro disponibilità e gentilezza, specialmente al gruppo del laboratorio di Perception e Manipulation per aver arricchito questa esperienza.

A Naiara, per non aver mai esitato nell'aiutarmi mentre ero lontano da casa.

Vorrei ringraziare anche il Dr. Gabriele Costante per il supporto nonostante la lontananza, ma soprattutto per avermi introdotto ed avermi fatto appassionare a quest'area di ricerca.

Grazie a tutta la mia famiglia per aver appoggiato e supportato ogni mia scelta, anche quelle non del tutto condivise.

Innanzitutto ai miei genitori, che non mi hanno mai chiesto nulla se non di provare a realizzare qualunque cosa io desiderassi. Per il senso di sicurezza che provate a trasmettermi ogni giorno, per l'educazione e i principi con cui mi avete cresciuto, per il costante insegnamento che "se tu puoi farlo, posso anche io", che mi hanno permesso di diventare ciò che sono oggi.

A mio fratello Alessio, alle mie nonne, agli zii e Sara, per aver condiviso con me gioie e preoccupazioni in tutti questi anni. Ai miei nonni, che sarebbero stati felici ed orgogliosi di vedermi oggi.

Agli amici di MAE, perché tutti questi anni sarebbero stati profondamente diversi senza di voi.

Infine a Marta, per esserci sempre stata anche nei momenti più difficili. Perché con te non ho paura dei cambiamenti e delle sfide che la vita potrà metterci davanti.