

Deep Networks for Analyzing Group Activity

Submitted in partial fulfillment of the requirements of Dual Degree
(Bachelor and Master of Technology) in Electrical Engineering

by

Ashish Goyal

Roll No. 12D070051

in the supervision of

Prof. Rajbabu Velmurugan



Department of Electrical Engineering
Indian Institute of Technology Bombay
India

June 28, 2017

Deep Networks for Analyzing Group Activity in Videos for Surveillance

Ashish Goyal

Abstract

A surveillance video can be analyzed by identifying activities at various levels of hierarchy- individual person, groups of persons and overall (or scene level). Most of the existing literature focuses on scene activity recognition and ignores multiple groups with different activities within a scene. Group level information can be employed for high-level applications such as abnormal activity detection and is important to understand the scene in its completeness.

We propose a deep hierarchical framework to analyze a video at three levels of hierarchy - individuals, groups of persons and overall scene. Unlike most of the existing approaches, our framework additionally discovers groups of people within a scene and identifies the corresponding group activity. We propose an objective function which learns the amount of pairwise interaction (compatibility) between any two persons in a scene. We also train our own pedestrian orientation recognition network whose performance is state of the art. As a minor contribution, we suggest post-processing steps which improve pedestrian detection for static cameras.

We evaluated our approach on standard datasets for group and scene activity, orientation estimation and pedestrian detection. The results of scene activity recognition are competitive with state of the art methods. Critically, unlike other approaches, our framework also detects groups of persons in a scene and the corresponding group activities with fair accuracy. Our orientation recognition network outperforms existing approaches and we also observe improvement in pedestrian detector performance due to the suggested post-processing algorithm. Our work contributed to Video Surveillance and Analysis project of National Center of Excellence in Technology for Internal Security (NCETIS).

Dedication

To you as a reader...

“Should I Support Vector Machines?”

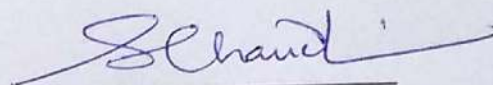
“The answer is very deep...”

Approval Sheet

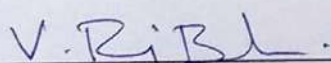
This dissertation entitled "**Deep Networks for Analyzing Group Activity**" by Ashish Goyal (12D070051) is approved for the degree of Dual Degree (Bachelor and Master of Technology) in Electrical Engineering with specialization in Communication and Signal Processing.



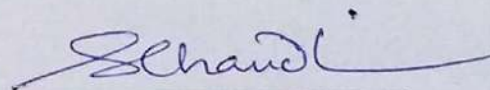
External Examiner
Dr. Udhav Bhosle



Internal Examiner
Prof. Subhasis Chaudhuri
Dept. of Electrical Engg., IIT Bombay



Guide
Prof. Rajbabu Velmurugan
Dept. of Electrical Engg., IIT Bombay



Chairman
Prof. Subhasis Chaudhuri
Dept. of Electrical Engg., IIT Bombay

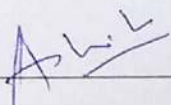
Date: June 23, 2017

Place: IIT Bombay

Declaration

I declare that this written submission is my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified my idea/data/fact/source in my submission. I understood that any violation of the above will cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has been taken when needed.

Signature: _____



Name of the student: _____

ASHISH GOYAL

Roll No.: _____

12D070051

Date: _____

27th June 2017

Acknowledgments

I would first like to thank my thesis advisor Professor Rajbabu Velmurugan of the Department of Electrical Engineering at IIT Bombay. He was always helpful in my troubles and provided great insights whenever I had doubts during research and writing. His guidance and motivation were critical in planning and completing my thesis.

I would also like to thank Neha Bhargava and Professor Subhasis Chaudhuri of the Department of Electrical Engineering at IIT Bombay for their consistent and valuable assistance during the later stages of my research. This work would not be in its current form but for their help and support.

I am thankful to National Center of Excellence in Technology for Internal Security (NCETIS) for sponsoring this project and providing excellent resources.

I am grateful to my parents and sister for their unfailing support during the course of my study. This would not have been possible without their continuous encouragement and motivation.

A special mention to everyone in the Vision and Image Processing lab. Working here really made me feel a VIP!

Lastly, but by no means least, big massive thanks to my best friend since childhood and all the friends I made at IIT Bombay for their emotional support. It was a privilege to share some of the best moments of my life with them.

Contents

1	Introduction	1
1.1	Events and Activities	1
1.2	Video and Camera	3
1.2.1	Location	3
1.2.2	Camera	3
1.3	Overview	3
2	Literature Survey	5
2.1	Pedestrian Detection	5
2.1.1	Region Proposals	6
2.1.2	Classification	6
2.1.3	Datasets	7
2.1.4	Evaluation	7
2.1.5	Challenges	8
2.2	Orientation Estimation	8
2.2.1	Datasets	10
2.3	Group Detection	10
2.3.1	Pedestrian Attributes	11
2.3.2	Learning Methods	11
2.4	Activity Recognition	12
2.4.1	Datasets	12
2.5	Summary	13
3	Methodology	14
3.1	Pedestrian Detection	14
3.1.1	Background Subtraction	14
3.1.2	Target Scale	15
3.1.3	Modified Non-Maximum Suppression (NMS)	16
3.2	Orientation Estimation	18

3.2.1	Modified Cross Entropy Loss function	18
3.3	Group Detection	21
3.3.1	Learning Pairwise Interaction Directly	24
3.4	Group and Scene Activity	26
3.4.1	Group Activity	26
3.4.2	Scene Activity	28
3.5	Summary	30
4	Results	33
4.1	Person Detection	33
4.2	Orientation Estimation	35
4.2.1	What layers to train?	36
4.2.2	Improved loss function	37
4.2.3	State of the art	37
4.3	Group Detection	42
4.3.1	Evaluation Metric for Group Detection	42
4.4	Group Activity Recognition	44
4.5	Scene Activity Recognition	45
4.6	Tuning the Network	47
5	Summary and Future Directions	49

List of Figures

1.1	Illustration of types of activities. Snapshots taken from popular datasets (see individual references for details)	2
1.2	A flowchart depicting steps involved in activity recognition and event detection. Except for tracking, we address each step mentioned in this figure.	4
2.1	The difference between classifiers and detectors. The image on the top depicts detection results and the image on the bottom shows classification result.	5
2.2	An overview of pedestrian detection with sliding window region proposals.	6
2.3	Types of error in pedestrian detection. False negatives 2.3a : Caused due to unusual illumination, clothing, camera height and tilt etc. False positives 2.3b : Confusion between a pedestrian and other objects like pole, traffic signs, vehicles etc. Bounding Box Shift 2.3c : Incorrect selection of correct BB among overlapping BB due to unreliable detector scores.	7
2.4	Selected frames underlining challenges of real surveillance videos.	9
2.5	Various approaches for group detection can be categorized in multiple ways. Our method uses data driven features and takes advantage of both trajectory and orientation. It also learns the importance of trajectories and orientations using training data, increasing its flexibility.	10
3.1	An example of background subtraction. The Left image is the computed background (I^b), the image in the center is an actual frame (I^t), and the right image is the computed silhouette (I^s). Clearly, in this case, silhouette detects pedestrians along with their shadows.	15
3.2	Calculation of scale factor $S(\mathcal{B})$ using projection of image coordinates of \mathcal{B} on to the ground plane.	16
3.3	A simple multilayer feedforward neural network architecture for learning affinity function.	23

3.4	Examples of different shaped groups possible in surveillance videos. DB-SCAN can discover these clusters based on pairwise affinities. Pointed shape of the circles illustrate orientation of the person.	24
3.5	Neural network architecture for group detection. We use 10 frames as our sequence length, primarily because annotations were provided at those intervals. Since activities are simple, observing sequences for longer durations does not help.	27
3.6	Neural network architecture for context unit. Pretrained ResNet50 is used as described in Section 3.2 with fixed weights.	28
3.7	Neural network for recognizing scene activity from groups.	30
3.8	Neural network for recognizing scene activity from individuals.	31
3.9	Summary of the complete model with various levels of hierarchy.	32
4.1	Qualitative comparison of baseline [20] before and after and post-processing steps. Left column contains the results before post-processing while the right column contains the results after post-processing. The first two rows indicate reduction in false positives and the remaining rows show decrease in missed detections.	34
4.2	Selected frames from the Parse27k [56] dataset used for training and testing the network.	38
4.3	Confusion plot and histogram of angular error for our <code>standard_loss_all_layers</code> model. For confusion matrices, the x -axis represents target (ground truth) class and the y -axis represents predicted (output) class. The x -axis in histogram depicts angular error in degrees ($^{\circ}$) and the y -axis stands for frequency. Class labels are explained in Table 3.1.	39
4.4	Confusion plot and histogram of angular error for our <code>total_loss_all_layers</code> model. For confusion matrices, the x -axis represents target (ground truth) class and the y -axis represents predicted (output) class. The x -axis in histogram depicts angular error in degrees ($^{\circ}$) and the y -axis stands for frequency. Class labels are explained in Table 3.1.	40
4.5	Confusion plot and histogram of angular error for our <code>total_loss_all_layers + aug</code> model. For confusion matrices, the x -axis represents target (ground truth) class and the y -axis represents predicted (output) class. The x -axis in histogram depicts angular error in degrees ($^{\circ}$) and the y -axis stands for frequency. Class labels are explained in Table 3.1.	41

4.6	Selected frames of results from collective activity dataset. Bounding boxes of same color represent same group. Last two frames indicate failure in group detection.	43
4.7	Confusion plot for our group activity recognition for the case when ground truth orientations and individual actions are unknown. For confusion matrices, x -axis represents target (ground truth) class and y -axis represents predicted (output) class.	45
4.8	Confusion plot for our scene activity recognition for the case when ground truth orientations and individual actions are unknown. For confusion matrices, x -axis represents target (ground truth) class and y -axis represents predicted (output) class.	46
4.9	A Plot of training accuracy after convergence against the number of hidden layers for group activity recognition network.	48

List of Tables

2.1	A comparison of commonly used datasets for pedestrian attributes. Boxes denotes the total number of bounding boxes in the dataset.	10
3.1	Table assigning orientations to angles.	18
4.1	Comparison of results on PETS 2009, TUD_Stadtmitte [41] and Towncentre [6] sequences. Our post-processing steps clearly indicate improvements over baseline [20]. Incorporating camera calibration data can further improve results. Values are average quality defined in Equation 4.1	33
4.2	Three variations of the model were investigated.	35
4.3	Model summary when all layers are being trained. 32 in the output shape represents the batch size.	35
4.4	Model summary when only last two fully connected layers are being trained. 32 in the output shape represents the batch size.	36
4.5	Summary of results for orientation estimation including all the models we investigated. <i>aug</i> represents data augmentation and <i>jitter</i> depicts PCA jittering.	36
4.6	Values of different metrics used to evaluate our approach. Values lie in the range of $[0, 1]$ and higher values indicate better performance.	42
4.7	Performance summary of group activity recognition under various settings. Naturally, any estimation error in a layer will affect performance of subsequent layer(s), as evident from these results.	44
4.8	Performance summary of scene activity recognition under various settings. Naturally, any estimation error in a layer will affect the performance of subsequent layer(s) as evident from these results.	46
4.9	Comparison of proposed approach with state of the art methods for scene and group activity.	47

Abbreviations

CCTV Closed Circuit Television

CNN Convolutional Neural Networks

fps Frames Per Second

HOG Histogram of Oriented Gradients

LSTM Long Short Term Memory

NMS Non-Maximum Suppression

RGB Red-Green-Blue

SIFT Scale Invariant Feature Transform

SVM Support Vector Machines

Notations

Unless stated otherwise:

- Scalars are represented by simple fonts - a, A, δ etc.
- Vectors and matrices are denoted by bold letters - $\mathbf{a}, \mathbf{A}, \mathbf{\Delta}$ etc.
- Styled fonts like $\mathcal{C}, \mathcal{S}, \mathcal{B}$ etc. are used for arbitrary sets.
- Double lined styled fonts denote sets of numbers. For example, the set of all real numbers is represented by \mathbb{R} .
- $\mathbb{R}^{a \times b \times c \times \dots}$ is a $a \times b \times c \times \dots$ dimensional space of real numbers.
- $\mathbf{A} \in \mathbb{R}^{a \times b \times c \times \dots}$ indicates A is an $a \times b \times c \times \dots$ dimensional array of real numbers.

Chapter 1

Introduction

We are living in a digital world, surrounded by electronic devices everywhere. These devices are designed to assist humans in carrying various tasks easily and efficiently. For example, imaging sensors like CCTV cameras enable monitoring of multiple locations from a centralized system, reducing the need of human presence. According to a survey by British Security Industry Authority (BSIA) [2], the number of CCTV cameras in the UK could be as high as one for every 11 people. With these numbers ever increasing, the human workforce is clearly inadequate to analyze these videos. Even though CCTVs are very useful for analyzing a scene after an event has happened, they are rarely used to detect or predict events.

Most of the surveillance videos can be divided into two categories:

- Involving humans - for e.g. classrooms, footpaths, hallways, shops, road crossings etc
- Not involving humans - for e.g. highways, parking lots, industries

In this work, we will solely focus on videos involving humans. The following sections elaborate on the scenarios and video recording settings we focused on in this study, followed by an overview of the problem.

1.1 Events and Activities

An event can be defined by various activities happening in a scene.

Example 1.1 In a football game, different players can carry out different activities. Suppose a striker is shooting the ball towards goal, the defenders are trying to tackle and the goalkeeper is trying to block the shot. Here, one team is attacking and the other team is defending. From the above information, we can infer that a "shot on target" event has happened.

It is interesting to note that in the above example, activities can be defined at various levels of hierarchy - individual player, group (team) and overall. We can divide activities into four types (refer to Figure 1.1 for examples):

1. Gestures and expressions
2. Actions
3. Pairwise interaction
4. Group activity



(a) Gesture [40]



(b) Action [54]



(c) Pairwise interaction [48]



(d) Group activity [11]

Figure 1.1: Illustration of types of activities. Snapshots taken from popular datasets (see individual references for details)

To focus on scenarios that generally arise in surveillance, we restrict the scope of this work to following class of actions:

1. Action - walking, standing, running, cycling, falling, bending etc.
2. Degree of pairwise interaction i.e. interaction or no interaction.
3. Group activity - walking together, talking, queuing, crossing a road, waiting together.

1.2 Video and Camera

There are numerous settings which can vary across surveillance videos - camera model, the number of cameras, video resolution, camera motion, the location of recording, proximity to the scene, crowd density and presence of objects like cars to list a few. Before going any further, it's critical to mention the types of videos we analyzed.

1.2.1 Location

We worked on both indoor and outdoor scenes in varying environments. In fact, as we will see later, the information about location actually helps in recognizing group activity. The videos have low crowd density, generally consisting of not more than 20 persons. Each person occupies only a small fraction of the frame and heights of persons vary in the range of 10% to 80% of the height of the frame. Although there's no restriction on illumination, it is important to have sufficient illumination so that the scene is clearly visible.

1.2.2 Camera

We worked with monocular videos recorded from RGB cameras with resolution around 720×640 pixels. The camera can be fixed, hand-held or even car mounted. However, fixed cameras do offer better performance due to lack of self (ego) motion. Ego-motion of hand-held and car-mounted cameras can be compensated using techniques like correlation or by tracking SIFT-like features. Cameras are placed at a minimum height of one meter. It is worth mentioning that lower camera heights lead to greater ambiguity in tracking vertical coordinates of objects. Videos are recorded at frame rates between 20 to 30 fps.

1.3 Overview

Analysis of video for surveillance is one of the most complex problems in computer vision. A bottom-up approach sees the problem as a combination of sub-problems at various levels of abstraction. Figure 1.2 illustrates one possible approach, depicting various facets of the problem. Except for tracking, we touch upon each of these facets.

- Low-level analysis - person detection, head detection, head and body tracking, body pose and gaze estimation, visual attention area etc.
- Mid-level analysis - identification of individual activity, person to person interaction, group discovery, recognition of group and scene activity, anomaly detection, personal space violation etc.

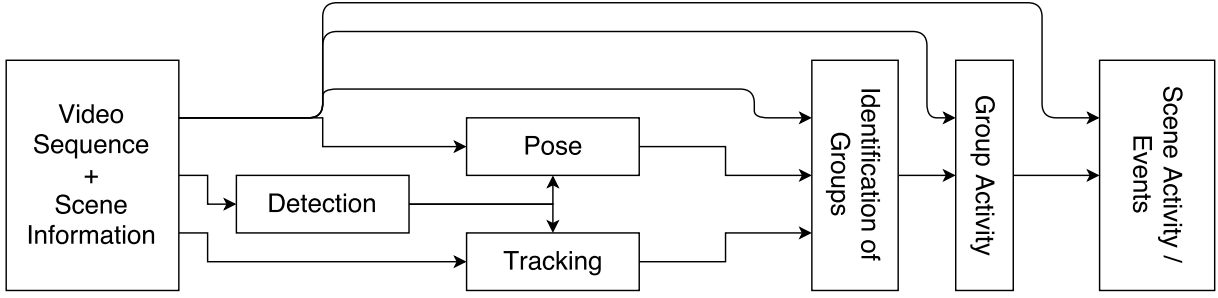


Figure 1.2: A flowchart depicting steps involved in activity recognition and event detection. Except for tracking, we address each step mentioned in this figure.

In this work, we explore various steps involved in analyzing a surveillance video, as illustrated in 1.2, except for target tracking. In the next two chapters, we will define the steps involved and review the relevant literature. Upon analyzing related work, we will discover drawbacks and limitations of existing approaches and address these issues in the remaining chapters. Specifically, we will employ deep learning methods for our analysis and compare their performance with state of the art. Our evaluation of these methods on standard datasets will provide insights into their usefulness, advantages and limitations.

Parts of this work were done in collaboration with Neha Bhargava, who is associated with Vision and Image Processing Lab, IIT Bombay. We are grateful to her for really helpful insights, support and annotations of course! Please refer to footnotes in Chapter 3 for details.

Chapter 2

Literature Survey

2.1 Pedestrian Detection

Object detectors find instances of objects in an image or video, for e.g. street signs, cars, pedestrians, faces, road etc. Object detectors are generally based on a binary classifier which discriminates between presence or absence of the object in an image. While classifiers tell only the presence or absence of an object in an image, detectors also localize the object in the image as illustrated in Figure 2.1.

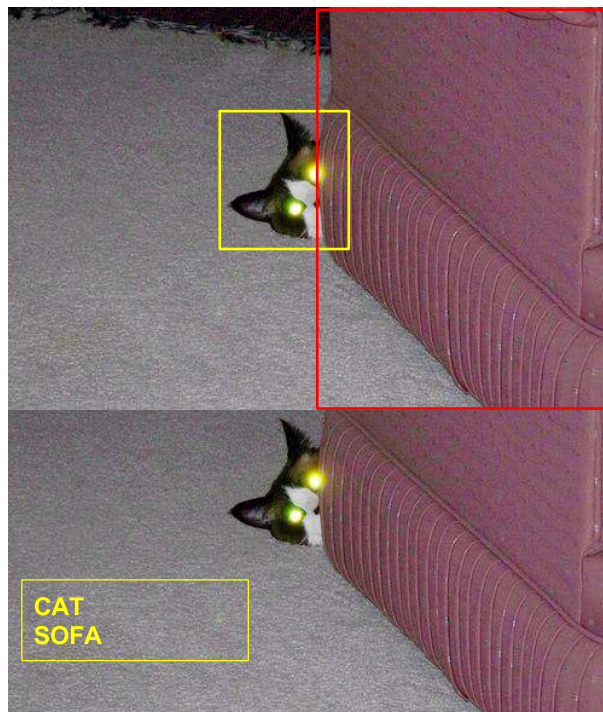


Figure 2.1: The difference between classifiers and detectors. The image on the top depicts detection results and the image on the bottom shows classification result.

2.1.1 Region Proposals

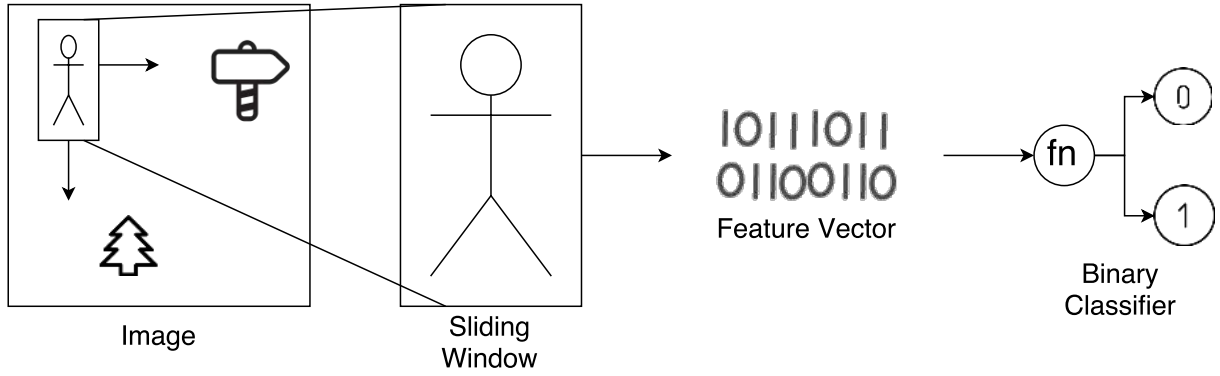


Figure 2.2: An overview of pedestrian detection with sliding window region proposals.

Typically, patches within an image (also called region proposals) are fed to a classifier which detects the presence of a pedestrian (see Figure 2.2). Patches can be sampled using various strategies and some of them are listed below:

1. Sliding a window across the image at various scales (please refer to [20] for details). While this method is slow as it generates a large number of region proposals, it ensures that object is never missed except when classifier fails.
2. Generating proposals from the image itself, for example using edges [61] or a neural network [46].
3. Generating proposals using a fast classifier (which is generally weak) with sliding window approach and refining them using a strong classifier (which is generally slow) [29].

2.1.2 Classification

If the speed is disregarded, the performance essentially depends on the strength of features and classifier in use. The features and classifiers have evolved over the years from simple handcrafted features and SVM to complex CNNs. For a detailed analysis and performance comparison, one can refer to an excellent survey by Benenson *et al.* [5]. Major features and classifiers used in pedestrian detection are listed below:

1. Traditional approaches use a combination of handcrafted features with classifiers. These features include Haar, HOG, edges, texture, color histogram, cosine transform coefficients, and optical flow. SVM, decision trees, boosted and bagged weak classifiers are most commonly used for classification.

2. In the deep learning regime, either simple CNNs are fully trained or complex CNNs pretrained on massive datasets are fine-tuned, to prevent overfitting. Since inference on CNNs is generally slow, a weak classifier is sometimes used to generate region proposals.

2.1.3 Datasets

Various datasets have been formed to organize the research in this area, to train detectors and evaluate their performance. Caltech pedestrian dataset [21] introduced in 2012, contains sequence of images containing pedestrians taken from a vehicle. INRIA pedestrian dataset [17] is a collection of images rather than a sequence.

2.1.4 Evaluation

Scenarios (see Figure 2.3) which can independently or collectively reduce the quality of detections are listed below:

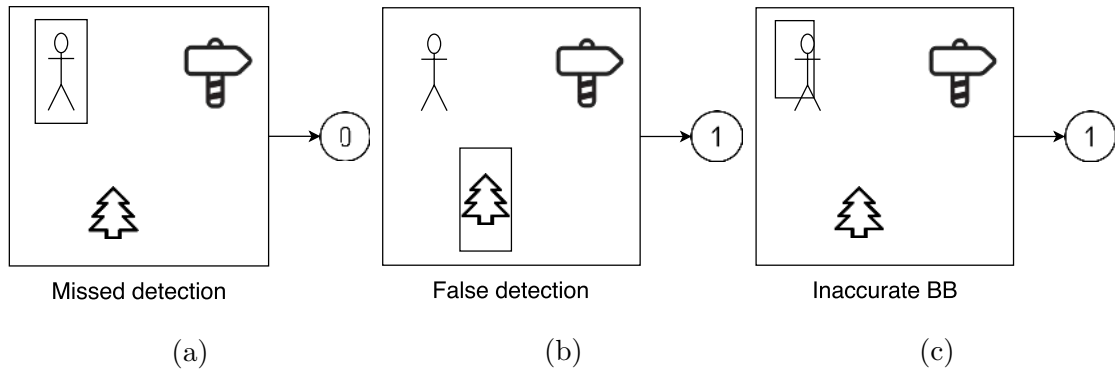


Figure 2.3: Types of error in pedestrian detection. False negatives 2.3a : Caused due to unusual illumination, clothing, camera height and tilt etc. False positives 2.3b : Confusion between a pedestrian and other objects like pole, traffic signs, vehicles etc. Bounding Box Shift 2.3c : Incorrect selection of correct BB among overlapping BB due to unreliable detector scores.

- **False Negatives** - Missed detections (see figure 2.3a) are the most common source of errors in detections. The appearance of pedestrians varies dramatically due to illumination, clothing, camera height, and tilt etc. This makes it hard for classifiers to generalize well. However, missing detections can be imputed using detection in neighboring frames in the context of videos.
- **False Positives** - Classifiers can easily confuse between a pedestrian and other objects like pole, traffic signs and vehicles to name a few (see figure 2.3b). Though

false detections are easier to handle in videos as they often tend to be outliers when multiple frames are considered.

- **Bounding Box Shift** - As evident from Figure 2.2, sliding window can be used to obtain region proposals from an image. Thus, multiple patches will indicate positive score around a pedestrian. Among all these overlapping windows, one with the highest score is chosen as the correct bounding box (BB). This process induces spatial shifts and inaccurate scaling in bounding box around the target (see figure 2.3c).

If the ground truth bounding box is known and denoted by B_{gt} , quality of detected bounding box B_{det} can be defined as:

$$Q = \frac{|B_{gt} \cap B_{det}|}{|B_{gt} \cup B_{det}|} \quad (2.1)$$

Where Q is the quality and $|\cdot|$ represents cardinality or area of the bounding box. \cap and \cup denote intersection and union respectively.

2.1.5 Challenges

To evaluate performance of pedestrian detectors in realistic surveillance videos, we analyzed a dataset obtained from Pune police in the context of detection using existing approaches [20, 5] (we are thankful to authors for providing implementations). Overall, we considered 7 different and challenging scenes, totaling 9233 frames at 25 fps. Figure 2.4 shows a montage of sample frames from selected sequences. Results obtained from this dataset outlined following challenges:

- Poor illumination, sudden changes in illumination.
- Foggy, rainy, hazy or dusty environment.
- Varying scale sizes of subjects.
- Low video resolution and poor video quality.

To tackle these challenges and improve performance of pedestrian detector, we suggest some priors to supplement existing approaches, specific to static cameras.

2.2 Orientation Estimation

Throughout this work, we define orientation as the orientation of the human body in space. Intuitively, orientation detection is critical to analyze interpersonal interactions



Figure 2.4: Selected frames underlining challenges of real surveillance videos.

and group formations. Orientation can also be loosely used as an indicator of visual focus of attention, finding direct applications in surveillance.

As discussed in Section 1.2.1, a pedestrian represents only a small fraction of an image, thus described by a small number of pixels. Such low resolution makes orientation estimation challenging and error-prone even for humans, constraining data labeling. Thus, instead of treating orientation as continuous quantity in $[0^\circ, 360^\circ]$, space is quantized in a number of bins (8 directions in this work). Under such constraints, orientation estimation essentially becomes multi-class classification problem with a cyclic order in classes.

It has been observed that the choice of classifiers and features mentioned in Section 2.1.2 work well for orientation estimation too [37, 43, 7, 57]. To further improve performance, time filtering of the orientation by combining information from velocity and appearance has been explored by Chen *et al.* [10].

Most approaches ignore the cyclic order of classes, thus, treating orientation estimation simply as a classification problem. As discussed in later chapters, we train our own CNN with modified cross entropy loss to account for this cyclic nature, out-performing state of the art in terms of mean angular error.

2.2.1 Datasets

Orientation of a pedestrian is one of several attributes describing a person, thus annotated datasets for orientation estimation are generally bundled with other attributes and called pedestrian attribute datasets. Table 2.1 presents a comparison between various attribute datasets that can be used for training orientation detector. Parse27k [56] contains a large set of training (50%), validation (25%) and test (25%) images. Moreover, video recorded on the same day are in same split, mitigating contamination across splits. For the aforementioned reasons, we train our orientation detector on this dataset.

Table 2.1: A comparison of commonly used datasets for pedestrian attributes. Boxes denotes the total number of bounding boxes in the dataset.

Dataset	Images	Boxes
Berkeley [9]	8,035	17,628
Parse27k [56]	9887	$\sim 27,000$
CRP [26]	20,999	27,454

2.3 Group Detection

Given a set of people in a scene, partitioning them into groups which follow some sociological reasoning is called group detection or discovery. While group detection has direct applications in surveillance to understand and monitor crowd behavior and detect (possibly predict) anomalies, it is also used for scene-level analysis. Although group detection has been studied for more than a decade[39], most of the approaches depend upon the specific definition of a group and the choice of social reasoning. Figure 2.5 presents an overview of group detection methods, with detailed explanation in subsequent paragraphs.

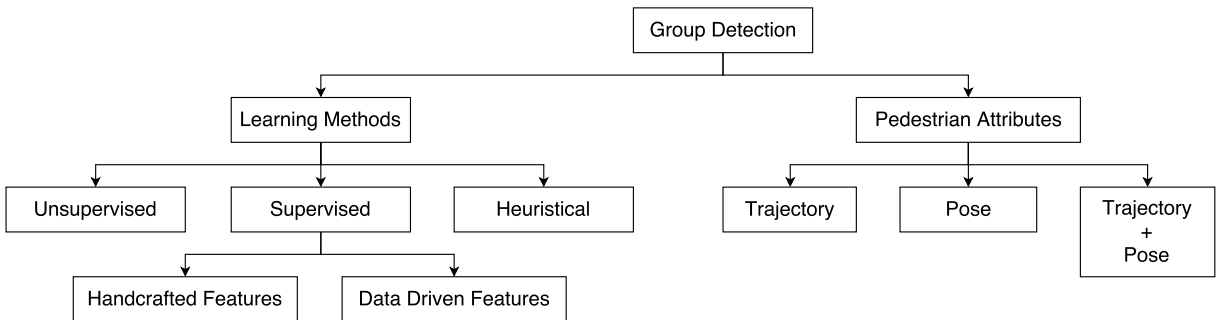


Figure 2.5: Various approaches for group detection can be categorized in multiple ways. Our method uses data driven features and takes advantage of both trajectory and orientation. It also learns the importance of trajectories and orientations using training data, increasing its flexibility.

2.3.1 Pedestrian Attributes

Benabbas *et al.* [4] track feature points using optical flow and use similarity of motion direction and spatial proximity to heuristically find groups. Jin *et al.* [31] define a pairwise affinity and use automatic fast density clustering to find group cores, which are further refined heuristically. Shao *et al.* [51] treat trajectories of individuals as a Markov chain and use transition matrix to infer about groups. Pellegrini *et al.* [44] use a conditional random field to jointly estimate trajectories and group. While these approaches work well in densely crowded dynamic scenes, they ignore orientation information which can be useful in sparsely crowded semi-stationary scenes.

In sparsely crowded semi-stationary scenes, where people can gather and talk, head or body orientation (orientation) can be used to find groups. Bazzani *et al.* [3] introduce inter-relation pattern matrix to represent pairwise interaction. Marco *et al.* [16] estimate F-formation [32] using Hough-voting. However, orientation estimates might not be reliable due to low resolution as mentioned in Section 1.2.1. Our approach uses orientation estimates as well as trajectories to detect groups and uses training data to decide the importance of each.

2.3.2 Learning Methods

Group detection approaches can be supervised, using structured learning [44], classifiers like SVM [59] and/or heuristics [4, 31, 51, 16, 3]. The features, however, are handcrafted (for e.g. proximity and similarity of velocity) and heuristics derived from sociological observations, limiting their applications to specific scenarios. In an interesting work, Solera *et al.* [53] determine contribution of multiple features through a Structural SVM and formulate the problem as supervised Correlation Clustering. However, the feature themselves are handcrafted, thus limiting the flexibility of the approach. Group detectors can also work in unsupervised fashion [23, 60], by extracting patterns hidden in the data using temporal smoothness. However, due to the complex nature of group dynamics, learning becomes difficult. Moreover, these approaches can not adapt to specific definitions of a group and the choices of social reasoning. Unsupervised models can not be informed about the intended grouping methodology by means of labeled data as they totally depend on priors. Although our approach requires annotated training data, it differs from state of the art due to two novel characteristics:

- It can adapt to various definitions of a group by learning the sociological reasoning from the training data itself.
- It only uses raw trajectories and orientation information, eliminating the need to

handcraft features.

2.4 Activity Recognition

Activity recognition is an active area of research due to its applications in video surveillance, crowd monitoring and event detection (read Chapter 1 for details). There are numerous approaches present in the literature on scene activity recognition. Some of the recent and interesting works are [12],[14],[13],[33],[38],[49],[50],[36],[24]. In [14], Choi *et al.* recognize the collective activity by extracting local spatio-temporal descriptors from people and the surroundings. They extend the algorithm in [13] by automatically capturing the crowd context and using it for classification. In [12], the authors go one step further and present a unified framework for target tracking and activity recognition. Ryoo and Aggarwal in [49] model a group activity as a stochastic collection of individual activities. The method proposed in [24] is based on multi-instance cardinality model with handcrafted features.

Recently, deep learning based methods have also been explored to recognize activities [30, 19, 18, 25]. Deng *et al.* proposed a deep model [18] to capture individual actions, pairwise interactions, and group activity. In another work [19], Deng *et al.* first estimate the individual and scene activities that are further refined using a message passing algorithm under a framework of a recurrent neural network. In [30], the authors proposed a two-staged LSTM model where the first stage captures individual temporal dynamics followed by scene activity recognition based on aggregated individual information.

Most of the existing approaches focus on scene activity recognition and ignore the fact that multiple groups with different activities are present in a video. Group level information can be employed for high-level applications such as abnormal activity detection and is important to understand the scene in its completeness. We build upon our group detector and detect group activities as well, along with scene activity.

2.4.1 Datasets

Creating annotated dataset for group and scene activity is a challenging task, since annotations have to be done at various levels, as illustrated in Figure 1.2. Choi and Savarese provide a dataset containing 44 sequences with annotated trajectories, pairwise interaction, and scene activity. We are grateful to Neha Bhargava ¹ for providing annotations of groups and group activity.

¹Neha Bhargava is associated with Vision and Image Processing Lab, Department of Electrical, IIT Bombay.

2.5 Summary

In this chapter, we explored several facets of surveillance video analysis. We defined the steps involved and reviewed the relevant literature. Upon analyzing related work, we discovered drawbacks of existing approaches for pedestrian estimation and group detection, a near complete void in literature for group activity recognition and scope of minor improvements in pedestrian detection. We address these issues using methods discussed in the next chapter. Subsequent chapters will validate the usefulness of our methods by comparing results with existing approaches. We will then summarize our findings with conclusions in the last chapter.

Chapter 3

Methodology

3.1 Pedestrian Detection

As discussed in 2.1.5, various challenges degrade the performance of pedestrian detectors. However, these detectors are designed for a general environment, assuming the following are unknown:

- Camera motion.
- Camera calibration.

We propose simple post-processing steps when the camera is static and its calibration is known.

3.1.1 Background Subtraction

Since most of the surveillance videos are obtained from cameras mounted at fixed points, the sensor system is static in nature, with the exception of PTZ cameras and mobile applications. Constraining camera to be static allows background estimation by taking the mode of images recorded for a long duration. Subsequently, background subtraction is done to obtain silhouette of foreground objects in each frame.

Let $I^t \in \mathbb{R}^{W \times H \times 3}$ be any RGB frame at time t , where W and H are width and height of the frame respectively. Let us also denote $I^b \in \mathbb{R}^{W \times H \times 3}$ as the background image, which is computed as:

$$I^b(x, y, c) = \underset{t}{mode}\{I^t(x, y, c) : t \in \{0, 1, \dots, T\}\} \quad (3.1)$$
$$\forall x \in \{0, 1, \dots, W - 1\}, y \in \{0, 1, \dots, H - 1\}, c \in \{R, G, B\}$$

Here, $\{0, 1, \dots, T\}$ is the time window for computing mode. T needs to be kept large enough for accurate background estimation. Foreground silhouette can be obtained by background subtraction across all color channels:

$$I^s = (I^b(., ., R) - I^t(., ., R))^2 + (I^b(., ., G) - I^t(., ., G))^2 + (I^b(., ., B) - I^t(., ., B))^2 \quad (3.2)$$

$I^s \in \mathbb{R}^{W \times H}$ is a gray-scale image. Higher the value of $I^s(x, y)$, more are the chances of pixel location (x, y) belonging to foreground. For any bounding box \mathcal{B} , we define a score $S_{\mathcal{B}}$ as:

$$S_{\mathcal{B}} = \frac{\sum_{(x,y) \in \mathcal{B}} I^s(x, y)}{|\mathcal{B}|} \quad (3.3)$$

Here $|\mathcal{B}|$ denotes the number of pixels in \mathcal{B} . This score $S_{\mathcal{B}}$ indicates whether \mathcal{B} is part of background or not. While this approach helps in detecting people, a major drawback is that it detects other moving objects like vehicles and animals. Other drawback is the detection of shadows along with the foreground object.



Figure 3.1: An example of background subtraction. The Left image is the computed background (I^b), the image in the center is an actual frame (I^t), and the right image is the computed silhouette (I^s). Clearly, in this case, silhouette detects pedestrians along with their shadows.

3.1.2 Target Scale

Let $h_{\mathcal{B}}$ be the height of a bounding box \mathcal{B} in image coordinates and $\frac{h_{\mathcal{B}}}{S(\mathcal{B})}$ be the height of \mathcal{B} in world coordinates. Here, $S(\mathcal{B})$ is a scale factor which converts height in image coordinates to the corresponding height in world coordinates. $S(\mathcal{B})$ is obtained by projecting image coordinates of \mathcal{B} on the ground plane using camera calibration and assuming zero elevation from the ground. The projected length of \mathcal{B} (see figure 3.2) is multiplied by the ratio of the height of camera sensor to the distance of camera sensor from the projected \mathcal{B} .

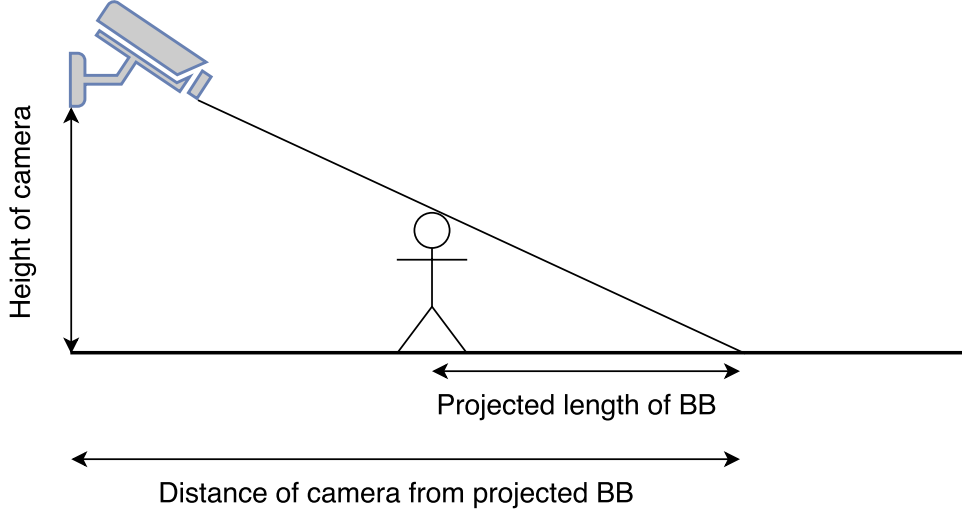


Figure 3.2: Calculation of scale factor $S(\mathcal{B})$ using projection of image coordinates of \mathcal{B} on to the ground plane.

We impose a prior on target scale since heights of most pedestrians lies within a certain range. Let us define a score function $T_{\mathcal{B}}$ as:

$$T_{\mathcal{B}} = ((\frac{h_{\mathcal{B}}}{S(\mathcal{B})} - H_{max})^+)^2 + ((\frac{h_{\mathcal{B}}}{S(\mathcal{B})} - H_{min})^-)^2 \quad (3.4)$$

Here, $(.)^+$ and $(.)^-$ are defined as:

$$x^+ = (x + |x|)/2$$

$$x^- = (x - |x|)/2$$

H_{max} and H_{min} respectively are maximum and minimum heights of most pedestrians in world coordinates. Even when camera calibration is unknown, a weak approximation can be followed by assuming $S(\mathcal{B}) = 1$ and setting H_{max} and H_{min} to maximum and minimum heights of most pedestrians in image coordinates anywhere in the image. The weakness of the approximation follows from the fact that $S(\mathcal{B})$ is independent of location, thus a larger variation in height has to be allowed.

3.1.3 Modified Non-Maximum Suppression (NMS)

As discussed in Section 2.1.1, detectors essentially consist of a classifier which gives probability (or confidence) $P_{\mathcal{B}}$ of a bounding box \mathcal{B} containing a pedestrian. To combine the information from background subtraction and target scale, we define a total score $L_{\mathcal{B}}$ as:

$$L_{\mathcal{B}} = \alpha S_{\mathcal{B}} + \beta T_{\mathcal{B}} + \gamma \log(P_{\mathcal{B}}) \quad (3.5)$$

Algorithm 1 Modified non-maximum suppression algorithm. Note that $A \setminus B = \{x \in A \mid x \notin B\}$ denotes relative complement. $T_{overlap} \in (0, 1)$ is the maximum amount of overlap (intersection over union) bounding boxes of two unique detections can have. Large value of $T_{overlap}$ introduces multiple detection for same pedestrian while small values tend to miss pedestrians standing close to each other. Intersection over union (or IoU) is defined as $|\mathcal{B}_i \cap \mathcal{B}_j| / |\mathcal{B}_i \cup \mathcal{B}_j|$. Step mentioned in Line 9 checks if two bounding boxes are overlapping more than threshold $T_{overlap}$. If so, the bounding box with lesser score ($L_{\mathcal{B}}$) is pruned off (Lines 10 to 14). These steps are repeated in a loop until no two bounding boxes overlap more than threshold $T_{overlap}$.

```

1: procedure NMS
2:   input:  $\mathcal{S} = \{(\mathcal{B}_0, L_{\mathcal{B}_0}), (\mathcal{B}_1, L_{\mathcal{B}_1}), \dots, (\mathcal{B}_N, L_{\mathcal{B}_N})\}$ ,
3:   top:
4:   if  $|\mathcal{B}_i \cap \mathcal{B}_j| / |\mathcal{B}_i \cup \mathcal{B}_j| < T_{overlap}$  for  $i \neq j$ ,  $\mathcal{B}_i, \mathcal{B}_j \in \mathcal{S}$  then return  $\mathcal{S}$ 
5:   else goto loop
6:   loop:
7:     counter = 0
8:     if  $i \neq j$  then
9:       if  $|\mathcal{B}_i \cap \mathcal{B}_j| / |\mathcal{B}_i \cup \mathcal{B}_j| > T_{overlap}$  then
10:        if  $L_{\mathcal{B}_i} < L_{\mathcal{B}_j}$  then
11:           $\mathcal{S} = \mathcal{S} \setminus (\mathcal{B}_i, L_{\mathcal{B}_i})$ 
12:          counter = counter + 1
13:        else
14:           $\mathcal{S} = \mathcal{S} \setminus (\mathcal{B}_j, L_{\mathcal{B}_j})$ 
15:          counter = counter + 1
16:     if counter > 0 then goto loop
17:     else return BB

```

Using $L_{\mathcal{B}}$, NMS can be applied to retain only one 'best' bounding boxes per detection. Algorithm 1 presents the steps involved in our modified NMS, which is the same as standard NMS¹ [20] except that we use total score $L_{\mathcal{B}}$ instead of classifier confidence $P_{\mathcal{B}}$. α , β and γ are manually chosen parameters. $S_{\mathcal{B}}$ and $T_{\mathcal{B}}$ are as described in equations 3.3 and 3.4 respectively.

From the remaining bounding boxes after NMS, one can choose appropriate ones using a threshold T_{ped} , which controls trade-off between miss rates and false positives (see Section 2.1.4).

¹<https://github.com/pdollar/toolbox/blob/master/detector/bbNms.m>

3.2 Orientation Estimation

As discussed in Section 2.2, most approaches ignore the cyclic order of classes, thus, treating orientation estimation simply as a classification problem. We train our own orientation estimator with modified cross entropy loss.

We used ResNet50 [27] pretrained on ImageNet dataset [47] as our choice of CNN. ResNet50 (50 layers) has fewer number of parameters than its deeper counterparts having up to 152 layers. ResNet50 (< 25 million parameters) has far lesser number of parameters than VGG16 (> 130 million) [52]. This is important since our dataset has $\sim 20,000$ images for training and overfitting looms for networks with a large number of parameters. Moreover, ResNet50 outperformed VGG16 in ImageNet competition as well. Since we have a moderate amount of training data with only eight classes, fine-tuning all the layers seems possible. Thus, we experimented with two settings:

1. Fine-tuning last two fully connected layers.
2. Fine-tuning all the layers.

3.2.1 Modified Cross Entropy Loss function

We quantize the space of orientations $[0^\circ, 360^\circ)$ into eight directions - right, right front, front, left front, left, left back, back, and right back. These eight classes possess cyclic order, and instead of classification accuracy, a more suitable choice of metric is average angular error measured as:

$$e_\theta = \frac{1}{N} \sum_{i=1}^N \min(|\theta(y_i) - \theta(\hat{y}_i)|, 360^\circ - |\theta(y_i) - \theta(\hat{y}_i)|) \quad (3.6)$$

Table 3.1: Table assigning orientations to angles.

Label	Orientation	Angle θ (in $^\circ$)
R	Right	$(-22.5, 22.5]$
RF	Right front	$(22.5, 67.5]$
F	Front	$(67.5, 112.5]$
LF	Left front	$(112.5, 157.5]$
L	Left	$(157.5, 202.5]$
LB	Left back	$(202.5, 247.5]$
B	Back	$(247.5, 292.5]$
RB	Right back	$(292.5, 337.5]$

Here, y_i is the predicted orientation and \hat{y}_i is the actual orientation. θ is a table (see Table 3.1) which assigns every angle in $[0^\circ, 360^\circ)$ to a class. Error metric defined in Equation 3.6 is unsuitable for use as loss function because it doesn't reflect classification

probabilities. Instead, we simply add extra penalties for misclassification to standard cross entropy loss when deviation from actual angle is more than 180° . We tried adding penalties for all misclassification instead of aforementioned criteria but training gets difficult as the objective function becomes too constraining. Consider standard cross entropy loss:

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{y \in \mathcal{C}} \delta(y, \hat{y}_i) \cdot \log(P_\Theta(y|x_i)) \quad (3.7)$$

$$\delta(x, y) = \begin{cases} 1, & \text{if } x = y. \\ 0, & \text{otherwise.} \end{cases}$$

\mathcal{C} is the set of all orientations (see Table 3.1). $P_\Theta(y|x_i)$ is the probability of class y i.e. output of the *softmax* layer for the neural network with weights Θ when image x_i is fed as input. Let us form a vector $\Delta_i = [\delta(1, \hat{y}_i), \delta(2, \hat{y}_i), \dots, \delta(8, \hat{y}_i)]^T$, also called one-hot encoding vector, since only the entry corresponding to correct class label will be one and rest will be zero. Additionally, let us form vectors $\mathbf{P}(\mathbf{x}_i) = [\log(P_\Theta(y = F|x_i)), \log(P_\Theta(y = RF|x_i)), \dots, \log(P_\Theta(y = RB|x_i))]^T$ and $\bar{\mathbf{P}}(\mathbf{x}_i) = [\log(1 - P_\Theta(y = F|x_i)), \log(1 - P_\Theta(y = RF|x_i)), \dots, \log(1 - P_\Theta(y = RB|x_i))]^T$, i.e. logarithm of the output of softmax layer in the CNN when input image is x_i and logarithm of its complement respectively. Equation 3.7 can be re-written in vector form as:

$$L = -\frac{1}{N} \sum_{i=1}^N \Delta_i^T \cdot \mathbf{P}(\mathbf{x}_i) \quad (3.8)$$

Here, $\mathbf{a} \cdot \mathbf{b}$ represents dot product of vectors \mathbf{a} and \mathbf{b} . Let \mathbf{D} be an 8×8 design matrix. We define additional penalty as:

$$K = -\frac{1}{N} \sum_{i=1}^N \Delta_i^T \mathbf{D} \cdot \bar{\mathbf{P}}(\mathbf{x}_i) \quad (3.9)$$

Matrix \mathbf{D} determines what misclassification will be penalized. For example, if the true class $\hat{y}_i = 1$, then $\Delta_i^T \mathbf{D}$ will pick the first row of \mathbf{D} . Therefore, the loss will be sum of entries of vector $\bar{\mathbf{P}}(\mathbf{x}_i)$ weighted by first row of \mathbf{D} . We can set the rows of \mathbf{D} to give more weight to corresponding wrong classes. Thus, if probabilities of wrong classes are high, penalty K will increase. Intuitively, \mathbf{D} must be circulant and symmetric matrix since the classes follow circular symmetry. We chose \mathbf{D} to be:

$$D = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The final objective function is a weighted sum of standard cross entropy and severe misclassification penalty:

$$J = - \left(w_1 \frac{1}{N} \sum_{i=1}^N \Delta_i^T \cdot \mathbf{P}(\mathbf{x}_i) + w_2 \frac{1}{N} \sum_{i=1}^N \Delta_i^T D \cdot \bar{\mathbf{P}}(\mathbf{x}_i) \right) \quad (3.10)$$

```

1 from keras.models import Model
2 from keras.layers import Dense, Input, Reshape
3 from keras.applications.resnet50 import ResNet50
4
5 resnet = ResNet50(weights='imagenet', include_top=False,
6                   input_shape=(224, 224, 3))
7 input_layer = Input(shape=(224, 224, 3))
8 resnet_features = resnet(input_layer)
9 resnet_features = Reshape(target_shape=(2048, ))(
10   resnet_features)
11 resnet_dense = Dense(1024, activation='relu')(resnet_features)
12 resnet_prob = Dense(8, activation='softmax')(resnet_dense)
13 orientation_resnet = Model(input=input_layer, output=
14   resnet_prob)

```

Listing 3.1: Keras [15] code to define the neural network architecture for orientation estimation. `include_top=False` in 5th line removes top two layers of the ResNet50. This code allows training of all layers.

3.3 Group Detection ²

We pose group detection as a supervised clustering problem. Consider a set of n people $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$. A clustering problem is to find K partitioning $\mathcal{C} = \{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_K\}$, $\mathcal{G}_i \subseteq \mathcal{P} \forall i \in \{1, 2, \dots, K\}$ such that:

$$\begin{aligned} \mathcal{G}_i \cap \mathcal{G}_j &= \phi \quad \forall i \neq j \\ \bigcup_{i=1}^K \mathcal{G}_i &= \mathcal{P} \end{aligned}$$

Here ϕ is a null or empty set and K is unknown. In words, partitioning is mutually exclusive and completely exhaustive. Additionally, members of same partitions should be 'close' to each other and any two members of different partitions should be 'far' apart. The definition of 'close' and 'far' can be condensed into an affinity function $f : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, where each element of \mathcal{P} is a d dimensional vector. In unsupervised clustering, no data is available to infer anything about f . Consider a different case, where N examples for such partitioning are given. In other words, a training set \mathcal{T} of N tuples $\mathcal{T} = \{(\mathcal{P}^1, \mathcal{C}^1), (\mathcal{P}^2, \mathcal{C}^2), \dots, (\mathcal{P}^N, \mathcal{C}^N)\}$ is given. Using this training data, we can come up with a better affinity function f such that it satisfies:

- Minimum intra-group affinity
- Maximum inter-group affinity

Consider an element $\mathbf{u} \in \mathcal{P}$, where \mathcal{P} has a known partitioning $\mathcal{C} = \{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_K\}$. Also let $\mathcal{G}(\mathbf{u})$ denote the group containing \mathbf{u} , i.e. $\mathcal{G}(\mathbf{u}) = \mathcal{A} \in \mathcal{C} | \mathbf{u} \in \mathcal{A}$. We define:

$$\begin{aligned} \alpha_{\mathbf{u}} &= \max_{\mathbf{v} \in \mathcal{P} | \mathbf{v} \neq \mathbf{u}, \mathbf{v} \in \mathcal{G}(\mathbf{u})} f(\mathbf{u}, \mathbf{v}) \\ \beta_{\mathbf{u}} &= \max_{\mathbf{v} \in \mathcal{P} | \mathbf{v} \neq \mathbf{u}, \mathbf{v} \notin \mathcal{G}(\mathbf{u})} f(\mathbf{u}, \mathbf{v}) \end{aligned}$$

Here $\alpha_{\mathbf{u}}$ is the maximum possible intra-group affinity and $\beta_{\mathbf{u}}$ is the maximum possible inter-group affinity for element \mathbf{u} . It is desirable to maximize $\alpha_{\mathbf{u}}$ and minimize $\beta_{\mathbf{u}}$ for each individual $\mathbf{u} \in \mathcal{P}$. Hence, we minimize an objective function J^i for i^{th} training tuple

²We are thankful to Neha Bhargava for her valuable inputs. She is associated with Vision and Image Processing Lab, Department of Electrical, IIT Bombay.

$(\mathcal{P}^i, \mathcal{C}^i)$, where J^i is defined as:

$$J^i = \sum_{\mathbf{u} \in \mathcal{P}^i} \max_{\mathbf{v} \in \mathcal{P}^i \mid \mathbf{v} \neq \mathbf{u}, \mathbf{v} \notin \mathcal{G}(\mathbf{u})} f(\mathbf{u}, \mathbf{v}) - \max_{\mathbf{v} \in \mathcal{P}^i \mid \mathbf{v} \neq \mathbf{u}, \mathbf{v} \in \mathcal{G}(\mathbf{u})} f(\mathbf{u}, \mathbf{v}) \quad (3.11)$$

As shown by Hornik [28] in 1991, using multilayer feedforward neural networks, continuous functions on compact subsets of \mathbb{R}^n can be approximated. Hence, neural networks can be used to represent many functions of interest. Let us bound the range of our affinity function to $f : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1]$. Since $[0, 1]$ is compact, f can be approximated using a multilayer feedforward neural network. Let us denote such a function by f_w , where w are weights of the neural network. w can be obtained by minimizing the objective function defined in Equation 3.12 using gradient descent as it is differentiable almost everywhere in its domain.

$$J^i = \sum_{\mathbf{u} \in \mathcal{P}^i} \max_{\mathbf{v} \mid \mathbf{v} \neq \mathbf{u}, \mathbf{v} \notin \mathcal{G}(\mathbf{u})} f_w(\mathbf{u}, \mathbf{v}) - \max_{\mathbf{v} \mid \mathbf{v} \neq \mathbf{u}, \mathbf{v} \in \mathcal{G}(\mathbf{u})} f_w(\mathbf{u}, \mathbf{v}) \quad (3.12)$$

Our neural network consists of two fully-connected hidden layers (see Figure 3.3 and Listing 3.2). Each layer has 32 neurons and *tanh* activation function. The prediction layer of the network is a single neuron with *sigmoid* activation function to ensure that the output is always in the range $[0, 1]$. *tanh* is hyperbolic tangent and *sigmoid* is a special case of the logistic function, defined as:

$$\begin{aligned} \tanh(z) &= \frac{(e^{(2z)} - 1)}{(e^{(2z)} + 1)} \\ \text{sigmoid}(z) &= \frac{1}{1 + e^{-z}} \end{aligned}$$

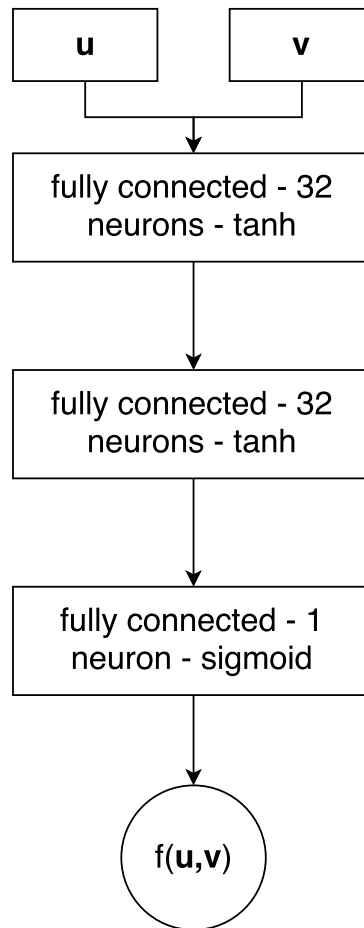


Figure 3.3: A simple multilayer feedforward neural network architecture for learning affinity function.

```

1 from keras.models import Model
2 from keras.layers import Input, Dense
3
4 n_hidden1 = 2*5
5 n_hidden2 = 2*5
6 f_len = 18
7
8 in1 = Input(shape=(f_len*2, ))
9 fc1 = Dense(n_hidden1, activation='tanh')(in1)
10 fc2 = Dense(n_hidden2, activation='tanh')(fc1)
11 out1 = Dense(1, activation='sigmoid')(fc2)
12 affinity_net = Model(inputs=in1, outputs=out1)

```

Listing 3.2: Keras code to define the neural network architecture. Note that input layer's size is `f_len * 2` since features of two persons are concatenated. Feature for each person consists of its bounding box's position, velocity and one-hot encoding of person's action and orientation.

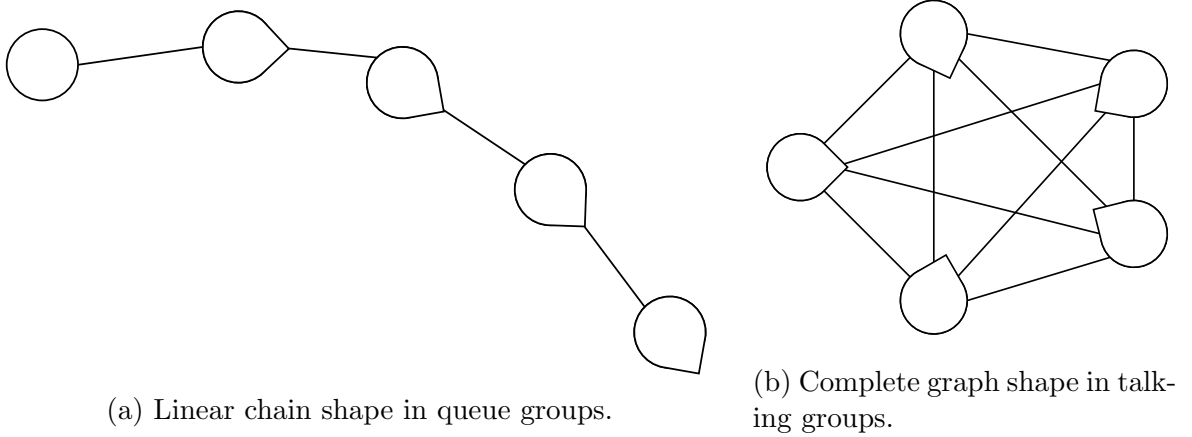


Figure 3.4: Examples of different shaped groups possible in surveillance videos. DBSCAN can discover these clusters based on pairwise affinities. Pointed shape of the circles illustrate orientation of the person.

Upon the completion of the optimization problem in Equation 3.12, the function f_w can be used to predict affinity between any two persons \mathbf{u} and \mathbf{v} . Suppose there are N people $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N \in \mathcal{P}$, where \mathcal{P} is a new set to be clustered. We compute the $N \times N$ affinity matrix \mathbf{A} such that $\mathbf{A}_{ij} = f_w(\mathbf{u}_i, \mathbf{u}_j) \forall i, j \in \{1, 2, \dots, N\}$. Clustering is performed on the affinity matrix \mathbf{A} using DBSCAN [22] algorithm. This algorithm has following properties essential for our task (see [22] for details):

- It can discover clusters of arbitrary shape. This is an important property as people can form groups in various shapes. For example, in *queues*, any person can have at most two neighbors. This gives rise to a linear chain shaped cluster (Figure 3.4a). However, in *talking* groups, each person may be interacting with everyone else. Thus, a complete graph shaped cluster is formed (Figure 3.4b).
- It doesn't need the number of clusters as a parameter, which is not known while detecting groups. This algorithm only uses a threshold on maximum distance beyond which two samples are not considered in the same group, which has an intuitive interpretation in terms of our affinity measure.

3.3.1 Learning Pairwise Interaction Directly

We formulated the group detection problem as supervised clustering. We proposed a solution involving training and testing phases. During training, we learn an affinity measure \mathbf{f}_w between two individuals. During testing, we use this measure to find the affinity matrix \mathbf{A} , which is used by DBSCAN algorithm to find the groups. Learning affinity measure \mathbf{f}_w becomes non-trivial when pairwise interactions are not available. Therefore, we formulated an optimization problem 3.12.

One can indeed learn pairwise interaction directly from annotations which might be even more informative than just the amount of interaction. For example, two individuals can stand side by side while waiting on a road crossing, they may face each other while talking or one may face the back of other while standing in a queue. As evident from these examples, such rich interaction information will definitely be extremely helpful for activity analysis. However, there are $\mathcal{O}(N^2)$ pairs to be annotated for N individuals, increasing the labeling effort multiple-folds. $\mathcal{O}(\cdot)$ denotes asymptotic order [8].

Choi *et al.* [11] did complete this mammoth task and their rich annotation was crucial in their approach. However, upon careful inspection, we concluded that their annotations were inconsistent. Numerous times, two individuals were labeled to be interacting when no obvious interaction was visible under any sociological reasoning. For this reason, we worked directly with group annotations.

Moreover, our approach has direct applications in any supervised clustering task. For example, image can be clustered by replacing fully connected network with a CNN which have proven to be remarkably powerful in image analysis. Similarly, sequences like speech or text can be clustered using LSTMs.

3.4 Group and Scene Activity ³

Once the groups have been discovered, the scene understanding starts to become clearer. Although behavior of different groups is interdependent, the inter-group interaction is rather weak and can be ignored, allowing us to analyze these groups independently. The scene can then be inferred from either using the group analysis information or from the individuals directly. However, analyzing different groups is important in itself as discussed in Section 2.4.

3.4.1 Group Activity

Once members of a group are known, we use their location, velocity, orientation and individual action information directly to recognize the group's activity. Consider a group $\mathcal{G} = \{\mathbf{u}_1, \mathbf{u}_1, \dots, \mathbf{u}_N\}$ of size N where $\mathbf{u}_1, \mathbf{u}_1, \dots, \mathbf{u}_N$ are feature vectors of N individuals, consisting of their location, velocity, orientation and individual action. We define a constant K , which denotes maximum number of persons we analyze in a group. Since there can be arbitrary number of individuals in a group, we consider three cases:

- $N = K$: Simply concatenate features of each individual forming a vector $\mathbf{g} = [\mathbf{u}_1 \frown \mathbf{u}_1 \frown \dots \frown \mathbf{u}_N]$, where $[\mathbf{x} \frown \mathbf{y}]$ is concatenation of vectors \mathbf{x} and \mathbf{y} . Note that $\dim(\mathbf{g}) = K \cdot \dim(\mathbf{u}_1)$.
- $N < K$: Concatenate features of each individual forming a vector $\mathbf{g} = [\mathbf{u}_1 \frown \mathbf{u}_1 \frown \dots \frown \mathbf{u}_N]$. Since $\dim(\mathbf{g}) < K \cdot \dim(\mathbf{u}_1)$, pad the vector \mathbf{g} with zeros to make $\dim(\mathbf{g}) = K \cdot \dim(\mathbf{u}_1)$.
- $N > K$: Randomly sample K individuals from the group \mathcal{G} , forming $\mathcal{G}' \subset \mathcal{G}$ with $|\mathcal{G}'| = K$. Here $|\mathcal{S}|$ is the cardinality of \mathcal{S} . Concatenate features of each individual in \mathcal{G}' to form a vector \mathbf{g} of dimension $\dim(\mathbf{g}) = K \cdot \dim(\mathbf{u}_1)$.

It is important to note that our approach is independent of ordering in a group and there is no precedence to one individual over other. However, one can construct problems, such as detection of events in a football game. For example, a *goal* event depends highly on activity of the person possessing the ball. In such cases, appropriate ordering should be chosen while concatenating the features of individuals.

We eliminate the need of handcrafted features by directly feeding vector \mathbf{g} to an LSTM. LSTMs are useful in processing sequences by extracting recurrent patterns using their memory cells. A major drawback of using LSTMs is the need of large datasets needed

³This work was done in collaboration with Neha Bhargava. She is associated with Vision and Image Processing Lab, Department of Electrical, IIT Bombay.

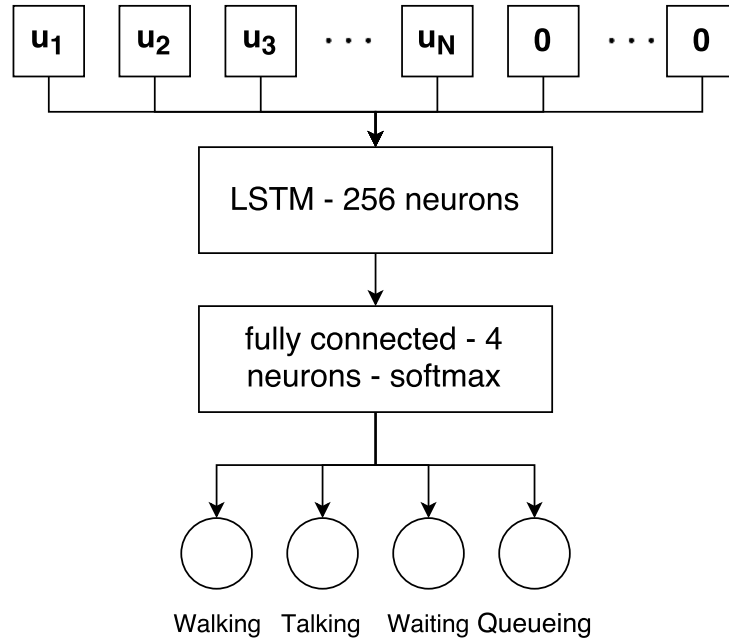


Figure 3.5: Neural network architecture for group detection. We use 10 frames as our sequence length, primarily because annotations were provided at those intervals. Since activities are simple, observing sequences for longer durations does not help.

to train them. Therefore we investigate the usability of LSTMs for activity recognition in the context of this work. Our model (see Figure 3.5 and Listing 3.3) consists of an LSTM with 256 hidden units. We feed the LSTM with group descriptor \mathbf{g} for 10 frames. State of the LSTM at every 10th frame is fed into a fully connected layer with 4 neurons and softmax activation for classification. Observing sequences for longer durations does not help because activities concerning the scope of this work are fairly simple and localized in time.

```

1 from keras.models import Model
2 from keras.layers import Input, LSTM, Dense, Masking
3
4 input_layer = Input(shape=(10, 180))
5 masked_in = Masking()(input_layer)
6 lstm1 = LSTM(256, activation='sigmoid', recurrent_activation=
    'tanh',
7 return_sequences=False)(masked_in)
8 fc1 = Dense(4, activation='softmax')(lstm1)
9 act_lstm = Model(inputs=input_layer, outputs=fc1)

```

Listing 3.3: Keras code to define the neural network architecture for group activity recognition. Masking layer is used to work with sequences of length less than 10.

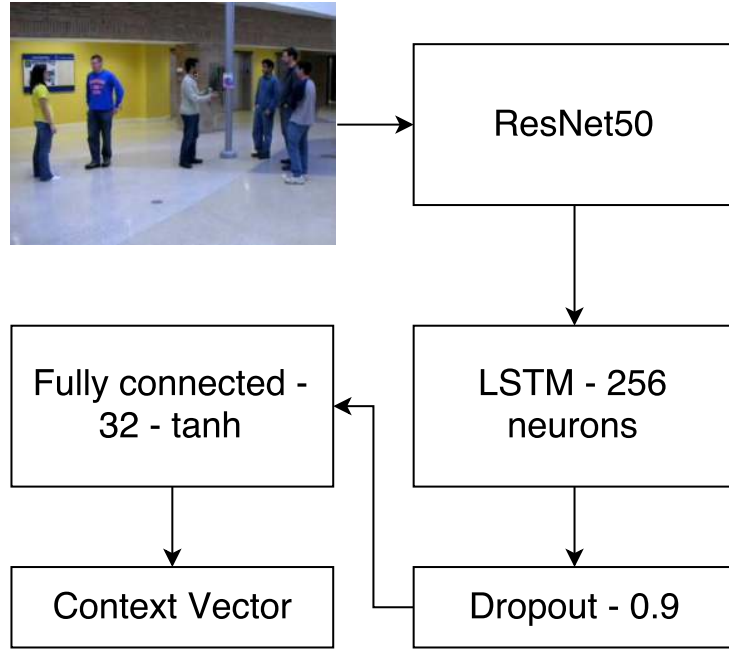


Figure 3.6: Neural network architecture for context unit. Pretrained ResNet50 is used as described in Section 3.2 with fixed weights.

3.4.2 Scene Activity

Intuitively, one should expect scene activity to be highly correlated with group activities. However, scene activity can be recognized directly from individuals too. Even though recognizing scene activity from group activities should be easier than directly deducing it from individuals, errors in group activity recognition can propagate and severely affect accuracy of scene analysis. In both cases, taking cues from scene context - location of the scene, presence of certain objects or places etc play a vital role in scene understanding. For example, walking on a road is actually a road crossing activity.

To fuse scene context with group/individual information, we use a context unit (see Figure 3.6 and Listing 3.4) to generate a vector of length 32. Firstly, each frame is fed into a pretrained ResNet50 (as mentioned in Section 3.2). Output of ResNet50 is fed to an LSTM with 256 neurons. To avoid over-fitting, we use dropout [55] with a rate of 0.9. This layer randomly drops off inputs from previous layer during training to avoid co-adaptation. High rates of dropout are needed since the dataset contains only 44 sequences. Output of the dropout layer is fed to a fully connected layer with 32 neurons and *tanh* activation. Output of this fully connected layer forms the context vector.

```

1 from keras.models import Model
2 from keras.layers import Input, LSTM, Dense, Masking, Dropout
3
4 resnet_layer = Input(shape=(10, 2048))
5 masked = Masking()(resnet_layer)
6 lstm1 = LSTM(256, activation='sigmoid', recurrent_activation=
    'tanh')(masked)
7 drop1 = Dropout(rate=0.9)(lstm1)
8 fc_context = Dense(16, activation='tanh')(drop1)

```

Listing 3.4: Keras code to define the neural network architecture for context unit. We precompute output of ResNet50 for all frames and use it as input in `resnet_layer`. `fc_context` is used as the context vector.

Scene activity from groups: We add the output of group activity recognition network 3.4.1 for all persons to form a vector of length 4. This vector is fed to a fully connected layer with 32 neurons and *tanh* activation. Output of this layer is concatenated with context vector and passed on as input to a classification layer with 5 neurons and *softmax* activation (see Figure 3.7). While it seems that deducing scene activity is easier from groups, errors in recognizing group activity may affect accuracy of scene activity recognition.

Scene activity from individuals: Similar to group analysis in Section 3.4.1, we concatenate individual descriptors (location, velocity, orientation and action) to form a scene descriptor of fixed length. We zero pad if there are fewer individuals than the fixed length and randomly sample if there are more. This scene descriptor is fed into an LSTM with 256 hidden units. Output of LSTM is passed on as input to a fully connected layer with 32 neurons, output of which is concatenated with context vector and supplied to a classification layer with 5 neurons and *softmax* activation (see Figure 3.8).

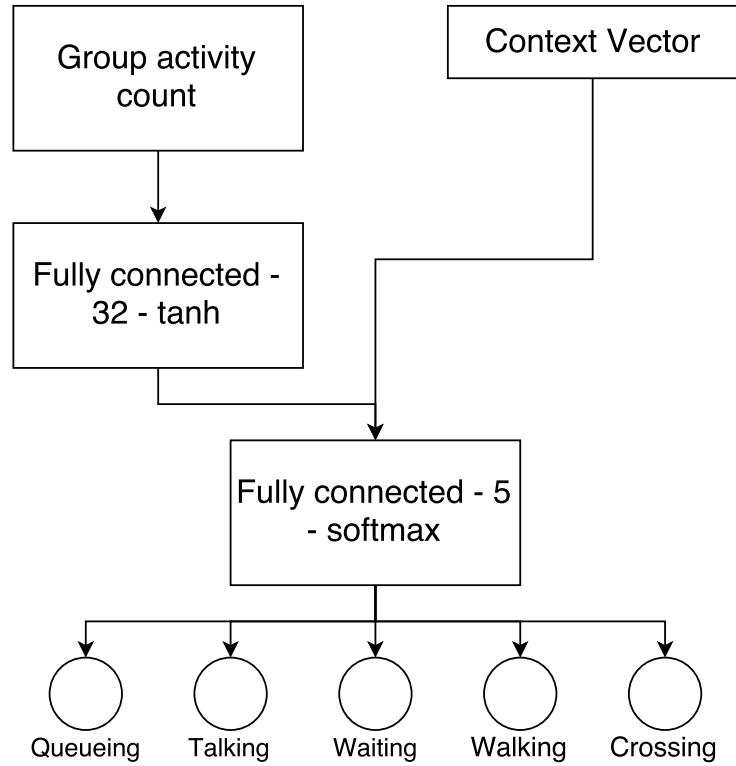


Figure 3.7: Neural network for recognizing scene activity from groups.

3.5 Summary

Understanding scene involves various stages as shown in Figure 3.9. The model can be summarized as follows. The inputs to our model are bounding boxes of the individuals (corner locations and color image) along with the complete scene image. The bounding boxes are fed to learn individual dynamics while the scene is supplied to learn the context. Firstly, we obtain the individual features which are passed to a clustering algorithm to discover groups in the scene. We train an LSTM based model to recognize group activities. To deduce scene activity, the model utilizes group activities and the scene context.

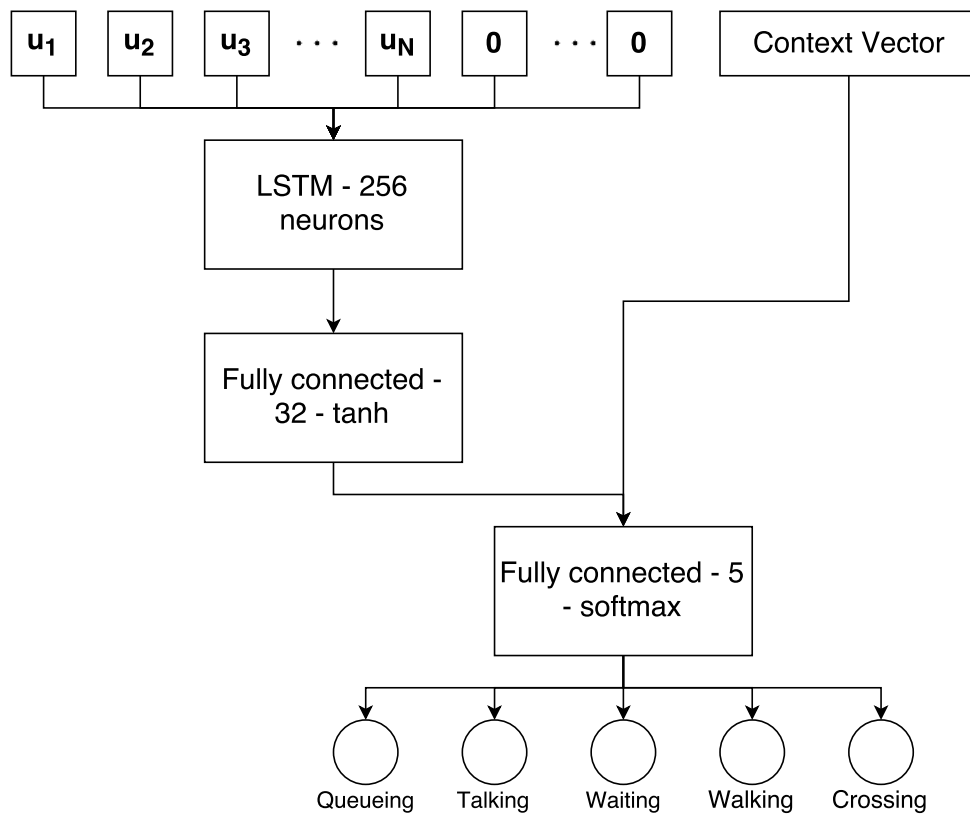
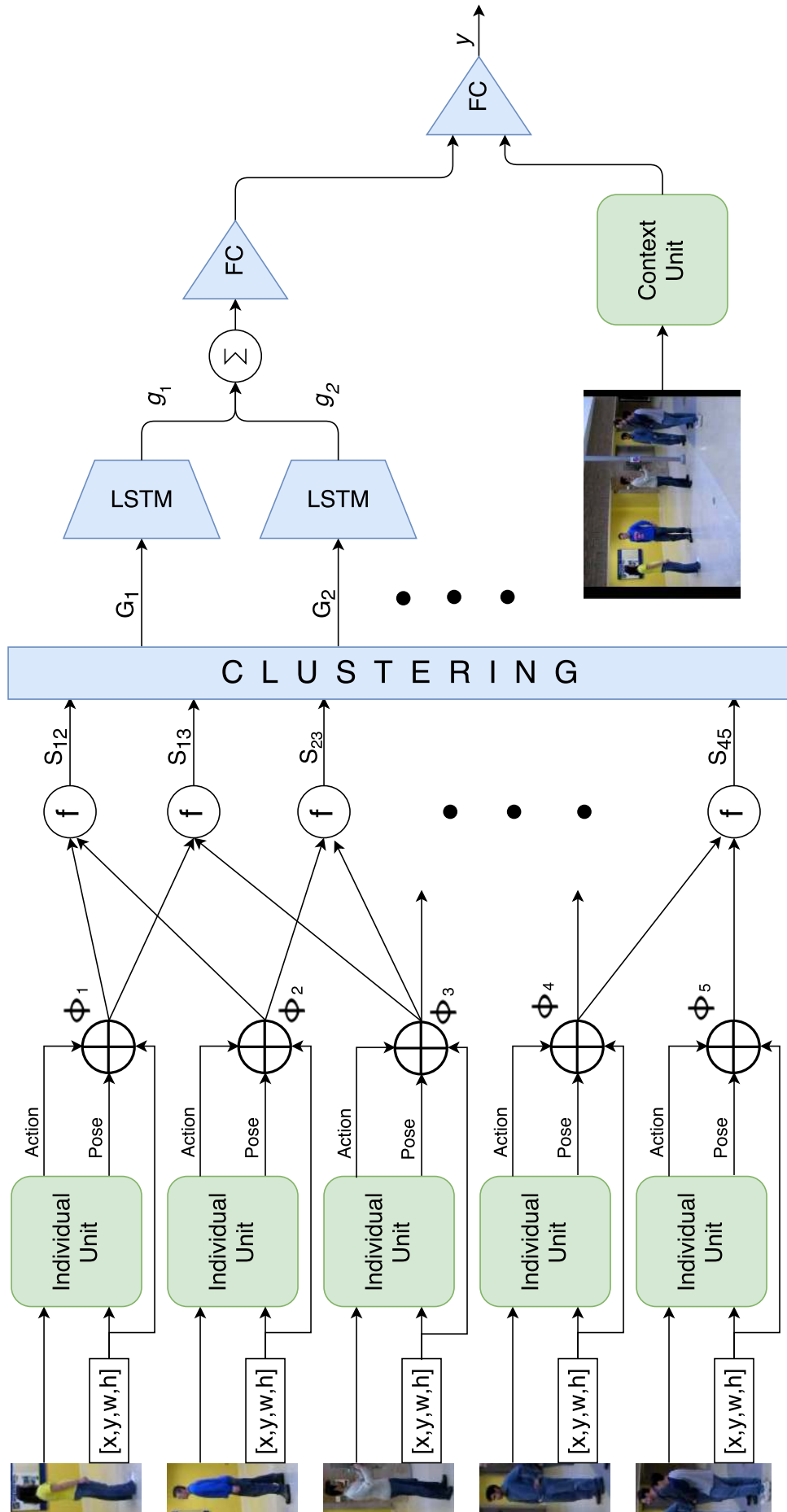


Figure 3.8: Neural network for recognizing scene activity from individuals.



32 Figure 3.9: Summary of the complete model with various levels of hierarchy.

Chapter 4

Results

4.1 Person Detection

In Section 2.1.4, we discussed three types of errors that frequently occur in person detection - missed detections, false detections and inaccurate bounding box localization. If the ground truth bounding box is known and denoted by B_{gt} , quality of detected bounding box B_{det} can be defined as:

$$Q = \frac{|B_{gt} \cap B_{det}|}{|B_{gt} \cup B_{det}|} \quad (4.1)$$

Q , as described above, incorporates both miss rate and bounding box quality. Since there's a trade-off between miss rate and false positives for any binary classifier, we investigate results of different methods by comparing quality Q , for equal false positive rates. We test our post-processing on three sequences - PETS 2009, TUD_Stadtmitte [41] and Towncentre [6] since ground truth is available and the camera is static. Results are summarized in Table 4.1. However, we do not use calibration data, which is further expected to improve results. Additionally, we also provide qualitative results on our own Pune dataset described in 2.1.5 in Figure 4.1.

Table 4.1: Comparison of results on PETS 2009, TUD_Stadtmitte [41] and Towncentre [6] sequences. Our post-processing steps clearly indicate improvements over baseline [20]. Incorporating camera calibration data can further improve results. Values are average quality defined in Equation 4.1

Sequence	Baseline	Post-processing
PETS09-S2L2	0.52	0.58
AVG-TownCentre	0.35	0.40
TUD-Stadtmitte	0.59	0.64

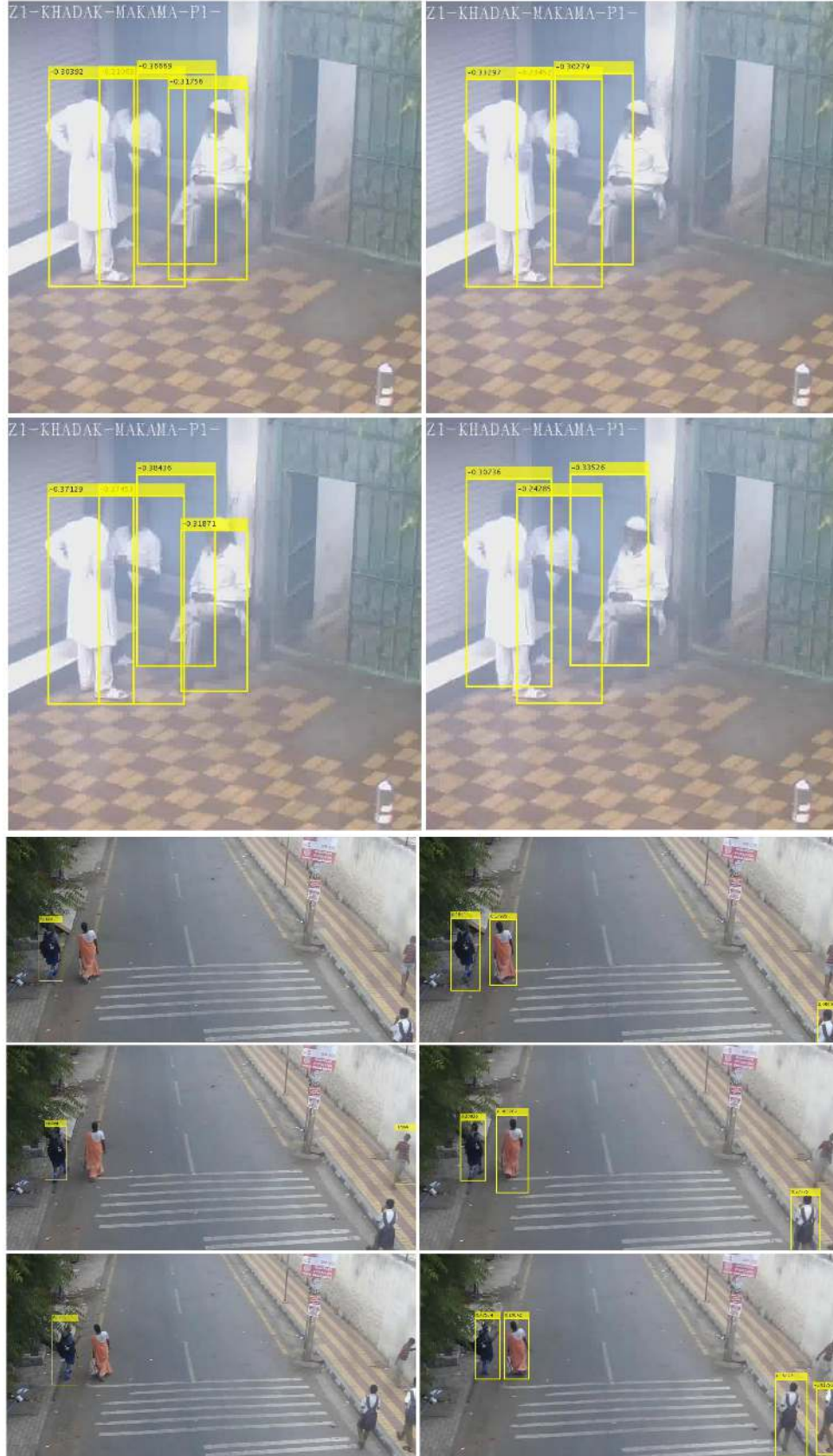


Figure 4.1: Qualitative comparison of baseline [20] before and after post-processing steps. Left column contains the results before post-processing while the right column contains the results after post-processing. The first two rows indicate reduction in false positives and the remaining rows show decrease in missed detections.

4.2 Orientation Estimation

We trained and evaluated our approach for orientation estimation on Parse27k [56] dataset described in Section 2.2. Figure 4.2 depicts selected images from the aforementioned dataset. We also compare variations of our approach as tabulated in Table 4.2.

Table 4.2: Three variations of the model were investigated.

Model Name	Description
total_loss_all_layers	Instead of standard cross-entropy loss, we used the objective defined in Equation 3.10 to train the network. All layers of the network were finetuned.
total_loss_fc_layers	Instead of standard cross-entropy loss, we used the objective defined in Equation 3.10 to train the network. Only last two fully-connected (fc) layers were finetuned.
standard_loss_all_layers	We used standard cross-entropy loss, while finetuning all the layers.

Tables 4.3 and 4.4 summarize different models used in our experiments. There are $\sim 27,000$ training images, which were resized to 224×224 to match input size of ResNet50. We preprocessed images (during training and testing) as mentioned in Keras [15] implementation¹ by subtracting mean [103.939, 116.779, 123.68] for RGB channels respectively. These values are channel-wise mean of ImageNet ILSVRC15 [47] training set images. We used a batch of 32 randomly sampled images from the dataset and fixed an initial learning rate of 0.0001. We used Adam [34] optimizer, eliminating the need of reducing learning rate since it adapts automatically. We train the model for 10 epochs, each epoch takes ~ 10 minutes on NVIDIA GTX 1080 GPU.

Table 4.3: Model summary when all layers are being trained. 32 in the output shape represents the batch size.

Layer	Output Shape	Number of parameters
Image Input	(32, 224, 224, 3)	0
ResNet50	(32, 1, 1, 2048)	23,587,712
Fully Connected (sigmoid)	(32, 1024)	2,098,176
Classification (softmax)	(32, 8)	
Total params		25,694,088
Trainable params		25,640,968
Non-trainable params		53,120

We compare our results with [56], where Parse27k dataset was originally proposed. We take their best performing model - Attributes Convolutional Net with hidden layers

¹https://github.com/fchollet/keras/blob/master/keras/applications/imagenet_utils.py

Table 4.4: Model summary when only last two fully connected layers are being trained. 32 in the output shape represents the batch size.

Layer	Output Shape	Number of parameters
Image Input	(32, 224, 224, 3)	0
ResNet50	(32, 1, 1, 2048)	23,587,712
Fully Connected (sigmoid)	(32, 1024)	2,098,176
Classification (softmax)	(32, 8)	
Total params		25,694,088
Trainable params		2,098,176
Non-trainable params		23,595,912

for each attribute (ACNH) trained separately for eight orientation estimation with data augmentation (random cropping, mirroring, and scaling) and PCA jittering as suggested in [35].

We use accuracy and average angular error (Equation 3.6) as our metrics of performance. Recall that average angular error is defined as:

$$e_{\theta} = \frac{1}{N} \sum_{i=1}^N \min(|\theta(y_i) - \theta(\hat{y}_i)|, 360^{\circ} - |\theta(y_i) - \theta(\hat{y}_i)|)$$

Table 4.5: Summary of results for orientation estimation including all the models we investigated. *aug* represents data augmentation and *jitter* depicts PCA jittering.

Model Name	Accuracy (in %)	Angular Error ($^{\circ}$)
ACNH [56]	68.2	-
ACNH + <i>jitter</i> + <i>aug</i> [56]	73.8	-
total_loss_fc_layers	36.8	67.4
standard_loss_all_layers	65.2	24.5
total_loss_all_layers	70.3	19.6
total_loss_all_layers + <i>aug</i>	74.5	16.6

4.2.1 What layers to train?

We compare two settings - total_loss_fc_layers and total_loss_all_layers (see Table 4.2). Results in Table 4.5 clearly indicate that training only the last two fully connected layers is not a good option. This happens because:

- The pre-trained network is optimized for recognizing objects in undistorted images. Since pedestrians generally have a 2:1 ratio of heights and widths, we expand these images along the width to meet input size requirements (224×224) of ResNet50. Thus, weights of ResNet50 need to be finetuned as well.

- The size of the fully connected layer is very large for the limited useful information that ResNet50 gives unless finetuned. This is confirmed by high training accuracy ($\sim 100\%$) as compared to poor validation and test accuracies ($\sim 40\%$).

4.2.2 Improved loss function

As expected, our improved loss function increases accuracies as well as angular error significantly as evident from Table 4.5 (compare `standard_loss_all_layers` and `total_loss_all_layers`). Standard cross entropy does not reflect ordinal nature in classes leading to poorer accuracies.

4.2.3 State of the art

Our best model - `total_loss_all_layers` + *aug* outperforms the best model (`ACNH` + *aug* + *jitter*) of Sudowe *et al.* [56]. Our base model (without data augmentation) also outperforms ACNH. Note that we do not employ PCA jittering in any of our models. Additionally, some layers of ACNH are pretrained on a semi-supervised task related to pedestrians [56] on a different data while we do not take advantage of any such data.



Figure 4.2: Selected frames from the Parse27k [56] dataset used for training and testing the network.

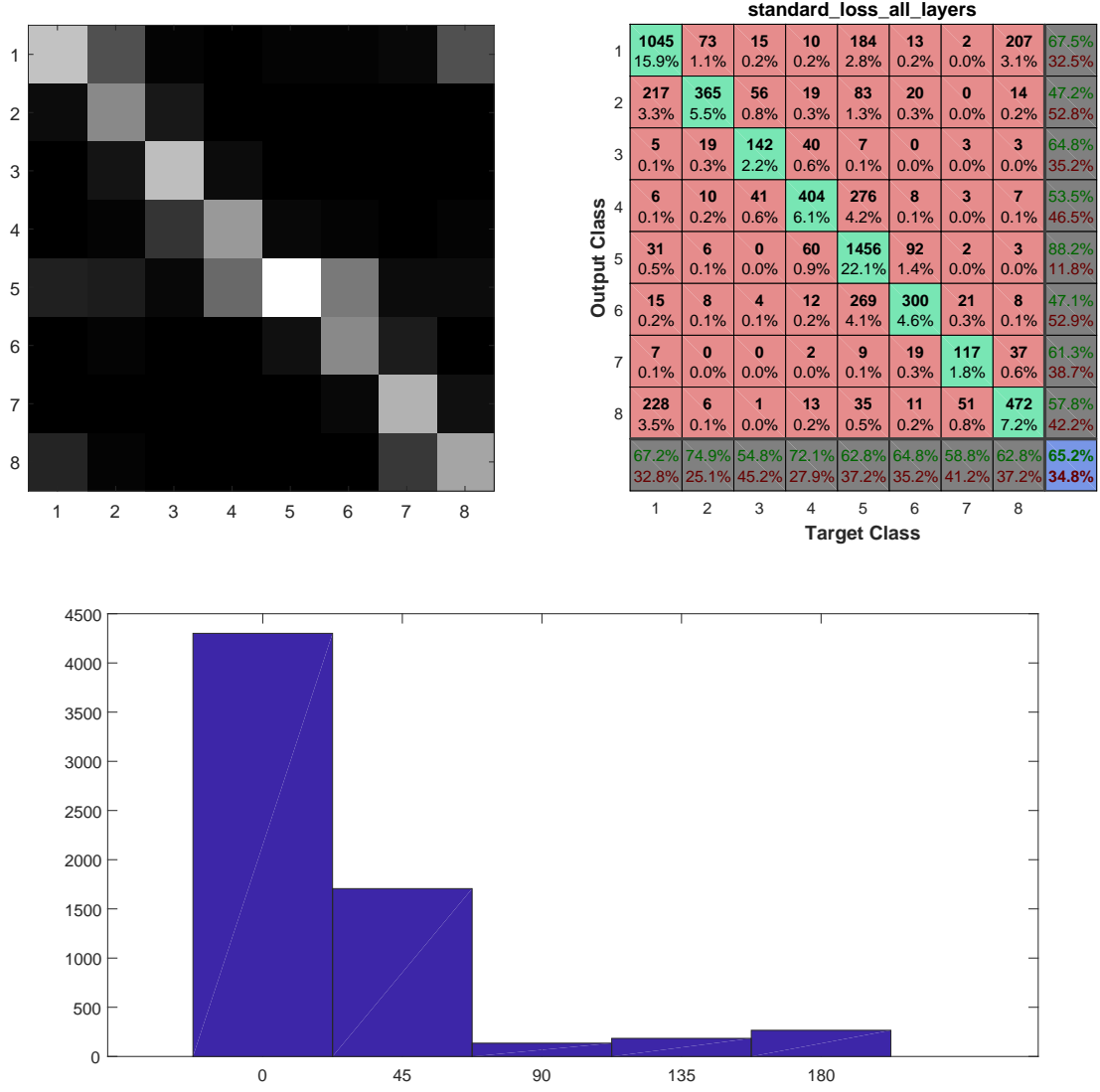


Figure 4.3: Confusion plot and histogram of angular error for our standard_loss_all_layers model. For confusion matrices, the x -axis represents target (ground truth) class and the y -axis represents predicted (output) class. The x -axis in histogram depicts angular error in degrees ($^{\circ}$) and the y -axis stands for frequency. Class labels are explained in Table 3.1.

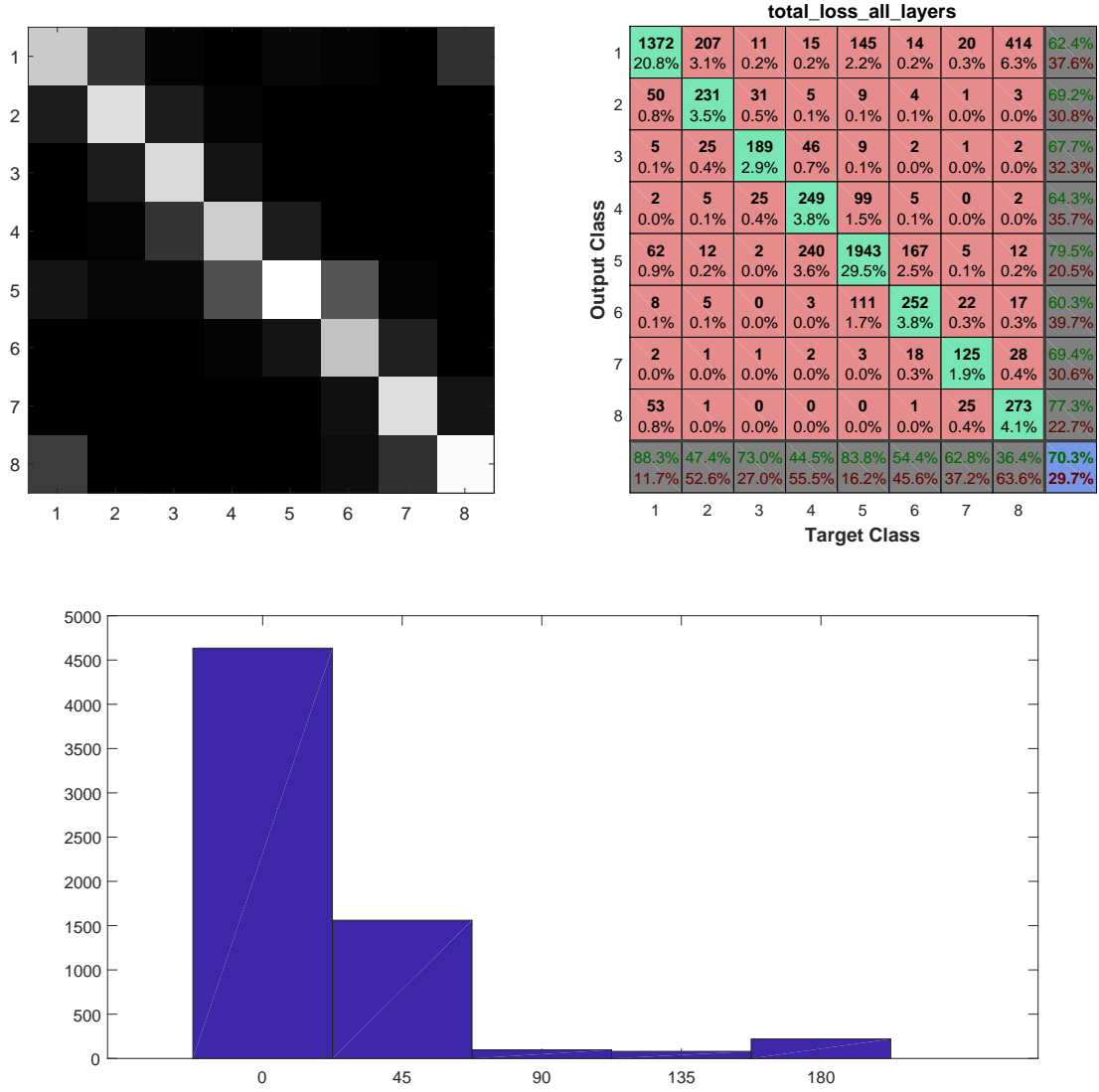


Figure 4.4: Confusion plot and histogram of angular error for our total_loss_all_layers model. For confusion matrices, the x -axis represents target (ground truth) class and the y -axis represents predicted (output) class. The x -axis in histogram depicts angular error in degrees ($^{\circ}$) and the y -axis stands for frequency. Class labels are explained in Table 3.1.

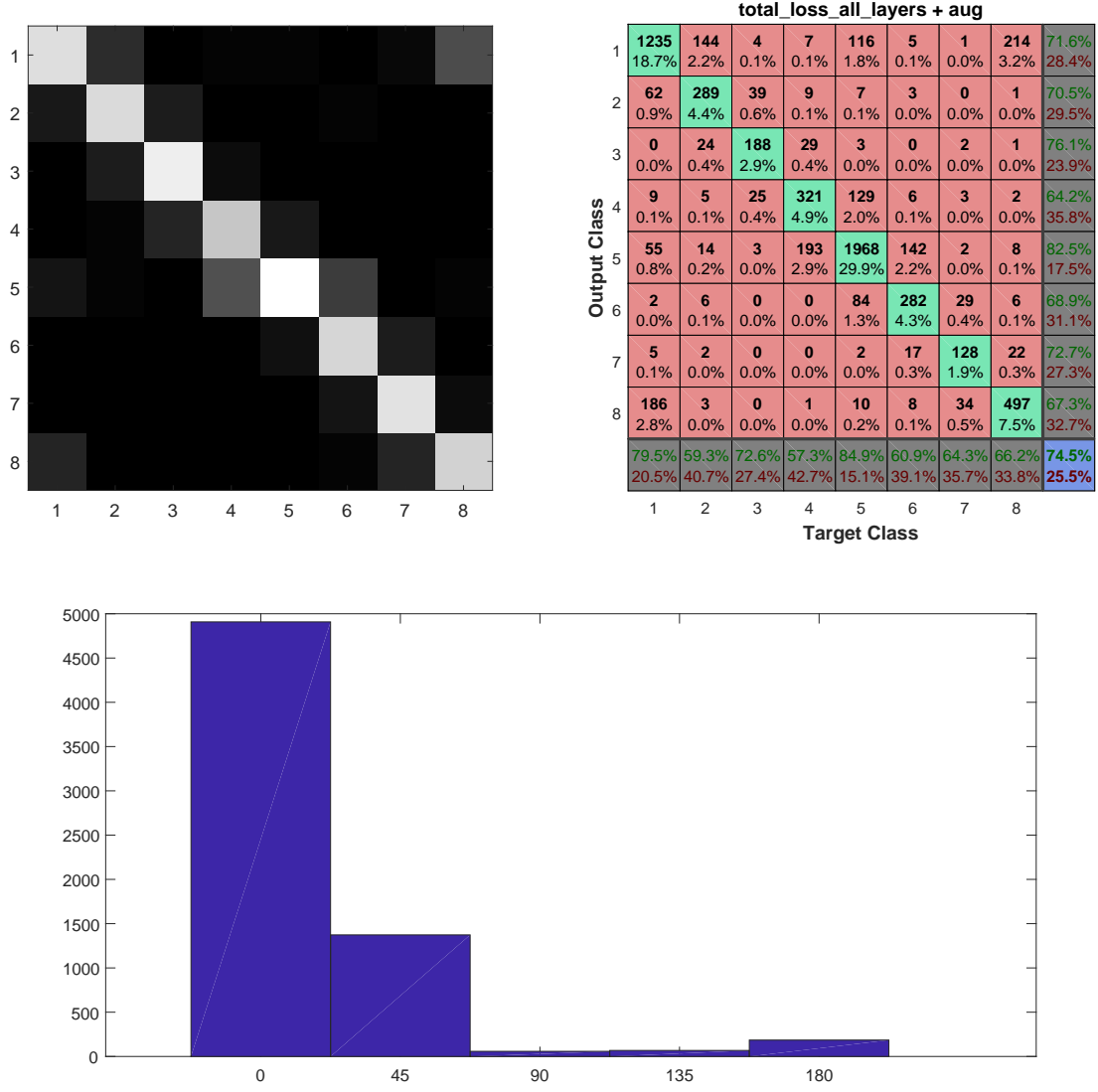


Figure 4.5: Confusion plot and histogram of angular error for our `total_loss_all_layers + aug` model. For confusion matrices, the x -axis represents target (ground truth) class and the y -axis represents predicted (output) class. The x -axis in histogram depicts angular error in degrees ($^{\circ}$) and the y -axis stands for frequency. Class labels are explained in Table 3.1.

4.3 Group Detection

As explained in Section 2.3.2, we explore group detection in scenarios where spatial location, velocity and orientation, all are necessary. Existing approaches work either on densely crowded dynamic scenes or sparsely crowded static scenes. Datasets like Collective Activity Dataset [11] having moderate crowds and semi-static scenes have not been looked upon from the perspective of group detection. We investigated our model using the aforementioned dataset and both qualitative and quantitative results are presented in upcoming paragraphs.

For each training video, we analyzed scene at every 10 frames. For each frame, we created one datum per person for each person in accordance with the objective function described in Equation 3.12. We formed batches of data, such that each batch contained every datum of a frame, leading to variable batch size. We used Adam [34] optimizer with an initial learning rate of 0.0001, eliminating the need of reducing learning rate since it adapts automatically. We stopped training after 5000 iterations since we observed a plateau in objective function value. The model takes ~ 100 seconds to train.

4.3.1 Evaluation Metric for Group Detection

1. **Normalized Mutual Information (NMI)** [58] - Inspired by information theory ideas and finds the mutual information between the two clustering outputs.
2. **Purity**[1] - Defined as the average percentage of the dominant class label in each cluster.
3. **Rand Index**[45] - Penalizes both false positive and false negative decisions during clustering.

We evaluated our approach quantitatively using above three metrics and the results are summarized in Table 4.6.

Table 4.6: Values of different metrics used to evaluate our approach. Values lie in the range of $[0, 1]$ and higher values indicate better performance.

NMI	Purity	Rand Index
0.82	0.88	0.89

Figure 4.6 illustrates some examples of success as well as failure cases in group detection. While the algorithm gives promising results, it fails to detect groups correctly for following possible reasons:

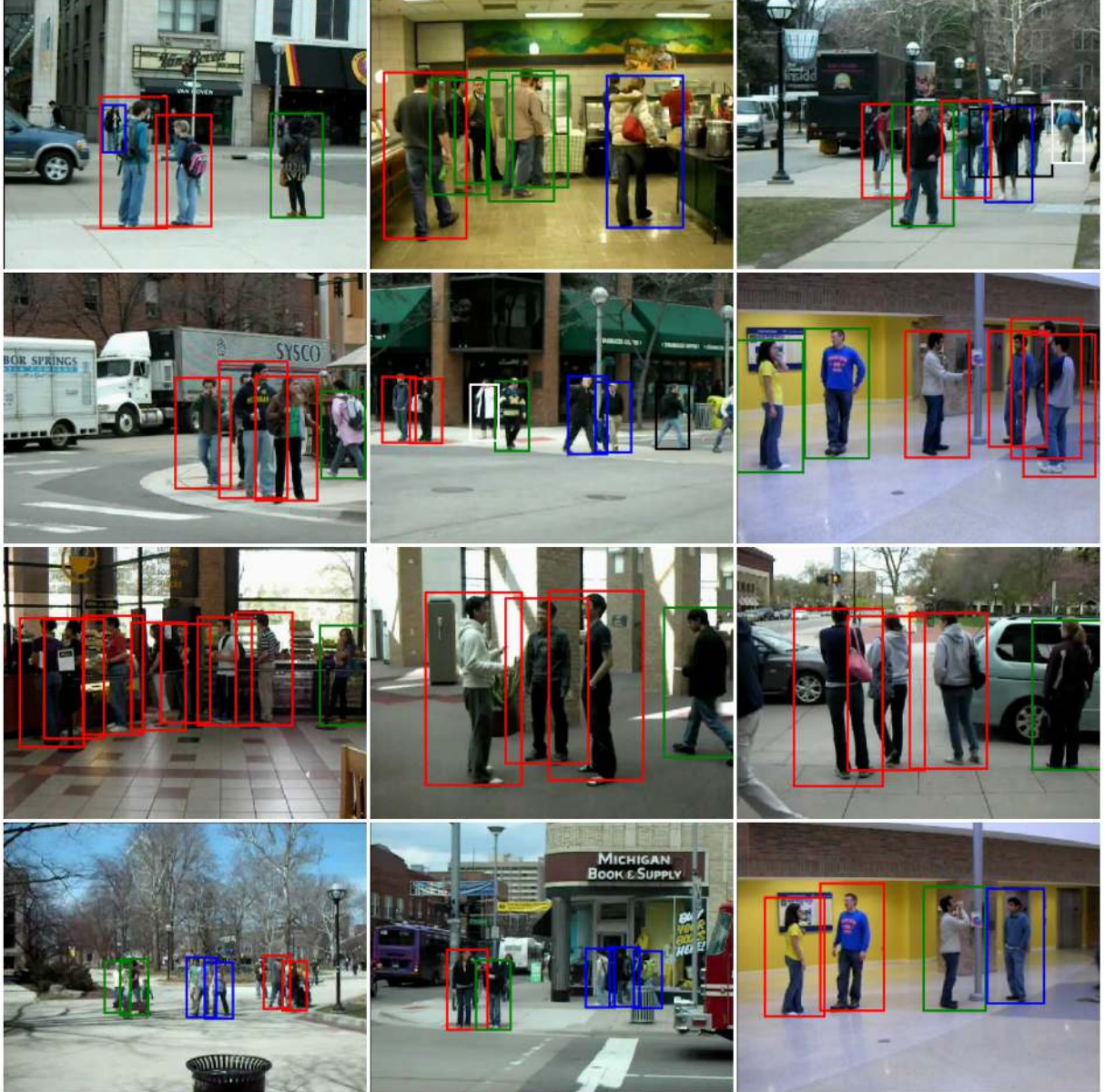


Figure 4.6: Selected frames of results from collective activity dataset. Bounding boxes of same color represent same group. Last two frames indicate failure in group detection.

- Errors from the previous stage, particularly orientation and action recognition distort the input of group detection algorithm, affecting its performance. In particular, our action recognition framework had a poor accuracy ($< 80\%$).
- Camera perspective skews ground plane severely with respect to the image plane. In other words, the ratio of length represented by a pixel in x coordinate to the corresponding length in y coordinate is nowhere close to 1. Thus, distance in image plane gives a very distorted picture of actual physical distance in the ground plane, which is fundamental to group detection.
- The dataset contains 44 video sequences only. A Small amount of data limits the learning of affinity function, affecting group detection performance.

4.4 Group Activity Recognition

In our experiments, we fixed the maximum number of people K possible in a group to be 10 (see Section 3.4.1). We observed that our group detection algorithm never clusters more than 10 persons in the same group. Standard categorical cross entropy was used as the loss function. We used Adam [34] optimizer with an initial learning rate of 0.001, eliminating the need of reducing learning rate since it adapts automatically. We used batches of size 512, where each datum of a batch contained descriptor of a group in a video frame and its corresponding activity. We train the model for ~ 100 epochs, where each epoch takes < 1 second to train on NVIDIA GTX 1080 GPU. The model takes ~ 100 seconds to train.

Table 4.7: Performance summary of group activity recognition under various settings. Naturally, any estimation error in a layer will affect performance of subsequent layer(s), as evident from these results.

Description	Accuracy (%)
Ground truth individual orientations and actions are known	86.2
Ground truth individual orientations are known but actions are unknown	82.1
Ground truth individual orientations and actions are unknown	80.2

Existing approaches focus on scene activities and therefore can't handle the cases where various groups perform different activities. Therefore, we can not provide a comparison. Figure 4.7 illustrates confusion plots of the obtained results on collective activity dataset [11] (see Section 2.4.1). To measure dependency of our model on previous layer(s), we investigate our model in three settings as summarized in Table 4.7.

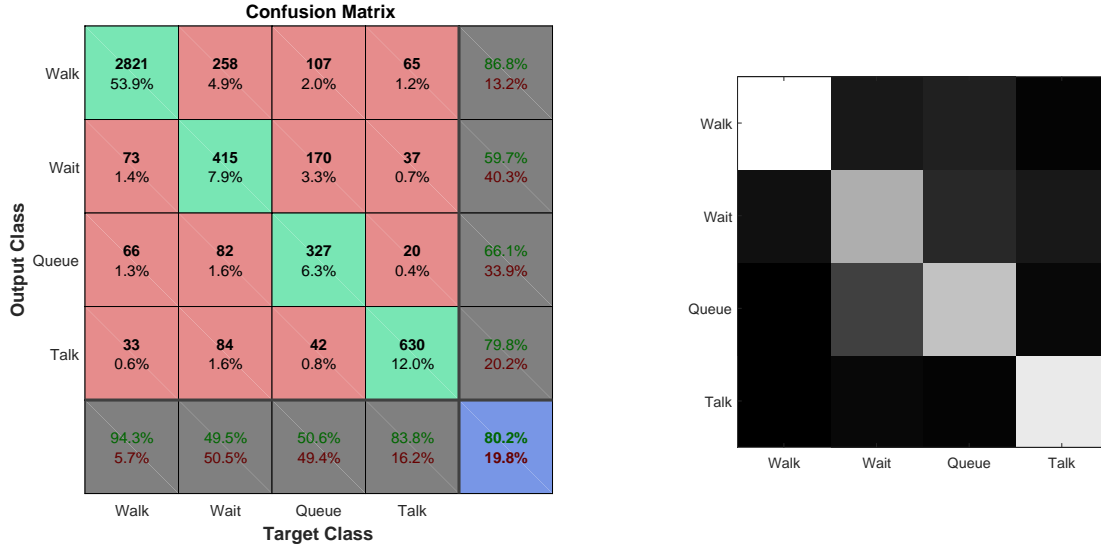


Figure 4.7: Confusion plot for our group activity recognition for the case when ground truth orientations and individual actions are unknown. For confusion matrices, x -axis represents target (ground truth) class and y -axis represents predicted (output) class.

We observe confusion in *wait* and *queue* activities. This happens possibly because the orientations of all members in a *waiting* group and a *queuing* group are the similar. Moreover, the members are static in both type of groups, making the distinction even more complex in absence of handcrafted features. Due to limited training data, the deep model is unable to learn discrimination in *waiting* and *queuing* groups. We hope that in future, more training data will be available, enabling the model to understand very complex groups (limitation!).

4.5 Scene Activity Recognition

In our experiments, standard categorical cross entropy was used as the loss function. We used Adam [34] optimizer with an initial learning rate of 0.001, eliminating the need of reducing learning rate since it adapts automatically. We used batches of size 128, where each datum of a batch contained descriptor of a video frame (see Section 3.4.2) and the corresponding scene activity. We train the model for ~ 100 epochs, where each epoch takes ~ 1 second to train. Our model takes ~ 100 seconds to train on NVIDIA GTX 1080 GPU.

In section 3.4.2, we considered two variants of our approach. When both ground truth individual orientations and actions are known, recognition of scene activity directly from individuals is 85.6 % accurate as compared to 88.9 % accuracy of scene activity recognition from group activities. When none of the ground truth individual actions

and orientations are known, these accuracies drop to 77.6 % and 80.2 % respectively. These results suggest that even though errors in recognizing group activities may affect performance of subsequent layer(s) (as we will see later in this section), learning scene activity directly from individuals is a very tough task.

Table 4.8: Performance summary of scene activity recognition under various settings. Naturally, any estimation error in a layer will affect the performance of subsequent layer(s) as evident from these results.

Description	Accuracy (%)
Ground truth individual orientations and actions are known	88.9
Ground truth individual orientations are known but actions are unknown	84.0
Ground truth individual orientations and actions are unknown	80.5

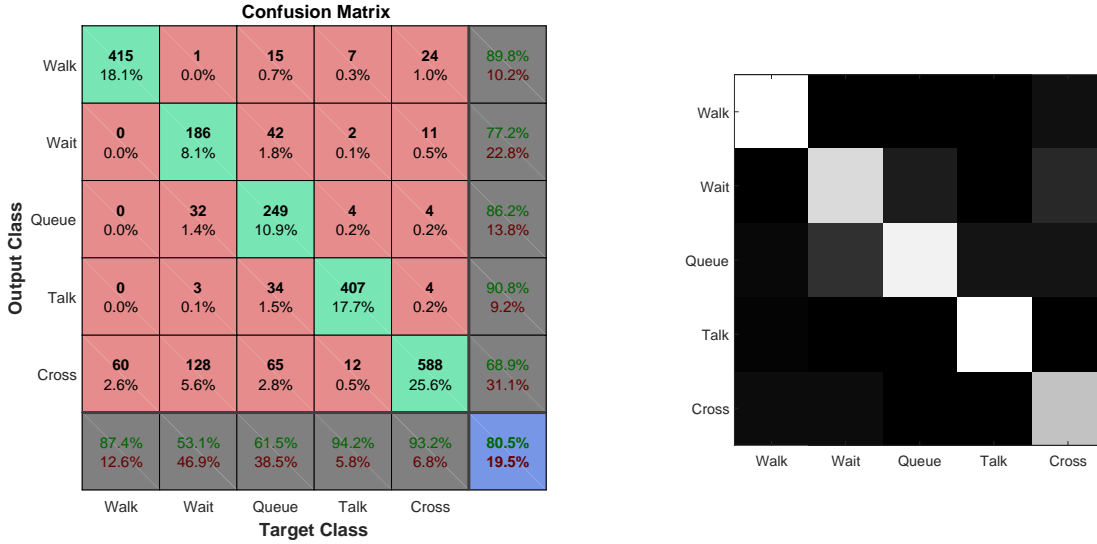


Figure 4.8: Confusion plot for our scene activity recognition for the case when ground truth orientations and individual actions are unknown. For confusion matrices, x -axis represents target (ground truth) class and y -axis represents predicted (output) class.

We compare the scene activity recognition performance with [19], [30], [18] and [24] on collective activity dataset 2.4.1. The average accuracies for comparison are mentioned in Table 4.9. The confusion matrix obtained from our method is shown in Figure 4.8. If we run the framework with true poses and true actions, the accuracy for scene activity reaches 88.9% and with true actions alone it goes to 84.0%. The average accuracy of our method is at par with state of the art methods. To measure dependency of our model on previous layer(s), we investigate our model in three settings as summarized in Table 4.8.

Method	Scene activity accuracy	Group activity accuracy
Cardinality kernel [24]	83.4%	-
Deep structured model [18]	80.6%	-
Structure Inference Machine [19]	81.2%	-
Hierarchical deep temporal network [30]	81.5%	-
Proposed method	80.5%	80.2%

Table 4.9: Comparison of proposed approach with state of the art methods for scene and group activity.

Upon observing the confusion matrix in Figure 4.8, we observe following major confusions:

- *Crossing* and *waiting* - Since our individual action estimation accuracy was $\sim 75\%$, *crossing* activity was misclassified as other static activities. However, due to the same context (road crossing in this example), it is misclassified as *waiting* activity frequently.
- *Crossing* and *walking* - The only difference between *crossing* and *walking* activity is the location where they are being performed. Since we feed context as a global cue, any scene where people are walking and there's a road is classified as *crossing*, regardless of whether people are actually crossing the road.(limitation!).
- *Wait* and *queue* - Since scene recognition layer in our model takes input from group activity layer, confusion in *wait* and *queue* is reflected here directly (see Section 4.4).

4.6 Tuning the Network

- **Activations** - The choice of activation function in various layers of a neural network is critical to its performance. We chose *tanh* activation for output of LSTM and *sigmoid* for its "gate layers" as is usually done in practice [42]. We found out empirically that other combinations of activation functions failed to converge. Fully connected hidden layers are activated using *tanh* function as we observed faster convergence as compared to *sigmoid*. Activations of classification (usually last) layer are fixed to *softmax* as the corresponding output resembles probabilities.
- **Layer Size** - Choosing size of hidden layer is important as it controls the trade-off between model strength and vulnerability to overfitting. Collective activity dataset has no validation splits. To mitigate overfitting while still selecting a strong enough

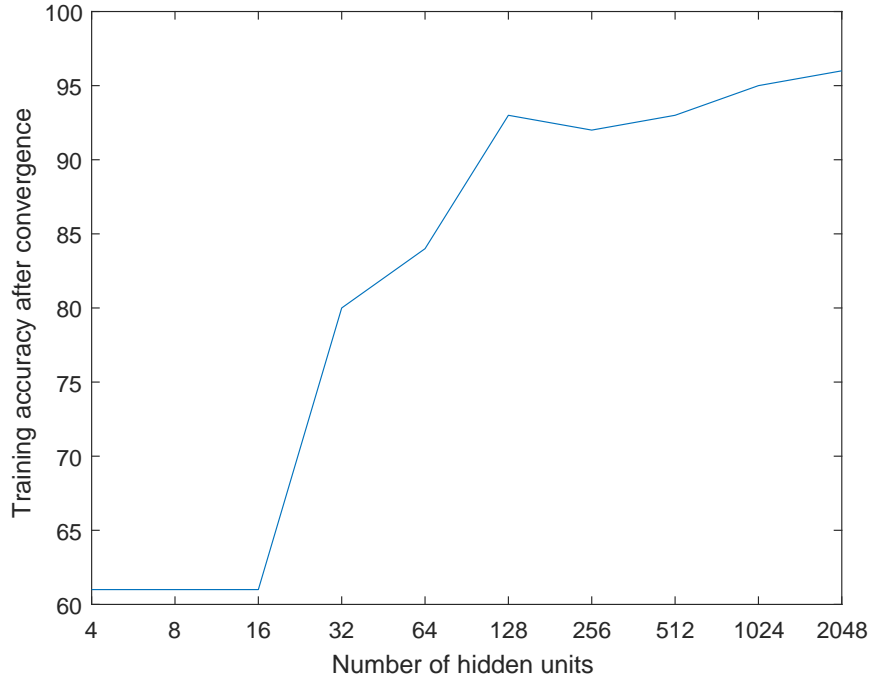


Figure 4.9: A Plot of training accuracy after convergence against the number of hidden layers for group activity recognition network.

network, we incremented layer size from 2 hidden units in multiples of 2, i.e. (2, 4, 16, 32, ...) and observed training accuracy. Initially, the model is weak, hence training accuracies are low. We stop increasing the size when we start observing a plateau in the plot of training accuracy against layer size. Figure 4.9 illustrates an example of this strategy while selecting the number of hidden units in group activity recognition LSTM.

- **Network Depth** - While the depth of a network increases its representation power, deeper architectures lead to increase in the number of parameters as well as the combinatorial complexity of optimization. We first fix the first layer's size using the strategy mentioned above. We then add an extra layer only when it significantly improves the training accuracy. The size of the second layer is then fixed and the process is repeated until no more layers are to be added. In practice, no unit (group detector, group activity classifier etc.) of our model was deeper than two layers.
- **Dropout** - We use dropout in scene recognition network. This is to prevent the model learning entirely from context (scene image). We select dropout rates using the same strategy as we employ while selecting the layer size. We start from dropout rate of 1 and decrease it in steps of 0.05 till we a plateau in training accuracy.

Chapter 5

Summary and Future Directions

In the previous chapters, we investigated the usefulness of neural networks for the task of group and scene activity analysis. We built a bottom-up approach and explored various facets of it. We evaluated the complete approach on a standard collective activity dataset. Results of various experiments suggested that using neural networks has the advantage of eliminating the need of handcrafted features and thus increasing flexibility to various scenarios. At the same time, we outperformed state of the art for orientation estimation and performed on par with state of the art in scene activity recognition. Additionally, our approach is able to handle multiple group activities within a scene.

Along with these advantages, we discovered some limitations of our model. In particular, due to limited training data (44 sequences in the dataset we evaluated upon), the deep model is unable to learn discrimination in *waiting* and *queuing* groups. There's a lack of datasets with annotations on several levels. While creating such datasets is difficult, collective efforts to take up such a task can be really helpful for training and evaluating various approaches. Another drawback of our model is that we use scene context as a global cue, and not for each individual locally. This was evident from the confusion between *crossing* and *walking* activities. Models, which use scene context globally as well locally can be explored in future.

While we analyzed group activities in this work, we did not touch upon other tasks in video surveillance. However, after developing a concrete understanding of the scene as our model does, an extension to other tasks in video surveillance should be easier. Let us consider anomalous activity detection for example. A typical scenario would be railway station ticket counter where the majority of the people are queued up. However, if someone leaves a suspicious package in between the queues, our model can easily detect it. This is because we know who is performing queue activity and which persons are performing other activities. Working to fill this semantic gap between group activity analysis and applications in video surveillance is an interesting future direction to work

upon.

Bibliography

- [1] C. C. Aggarwal. “A human-computer interactive method for projected clustering”. In: *IEEE Transactions on Knowledge and Data Engineering* 16.4 (2004), pp. 448–460. ISSN: 1041-4347. DOI: [10.1109/TKDE.2004.1269669](https://doi.org/10.1109/TKDE.2004.1269669).
- [2] David Barrett. *One surveillance camera for every 11 people in Britain, says CCTV survey*. 2013. URL: <http://www.telegraph.co.uk/technology/10172298/One-surveillance-camera-for-every-11-people-in-Britain-says-CCTV-survey.html>.
- [3] Loris Bazzani et al. “Analyzing Groups: A Social Signaling Perspective”. In: *Video Analytics for Business Intelligence*. Ed. by Caifeng Shan et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 271–305. ISBN: 978-3-642-28598-1. DOI: [10.1007/978-3-642-28598-1_9](https://doi.org/10.1007/978-3-642-28598-1_9). URL: http://dx.doi.org/10.1007/978-3-642-28598-1_9.
- [4] Yassine Benabbas, Nacim Ihaddadene, and Chaabane Djeraba. “Motion Pattern Extraction and Event Detection for Automatic Visual Surveillance”. In: *EURASIP Journal on Image and Video Processing* 2011.1 (2010), p. 163682. ISSN: 1687-5281. DOI: [10.1155/2011/163682](https://doi.org/10.1155/2011/163682). URL: <http://dx.doi.org/10.1155/2011/163682>.
- [5] Rodrigo Benenson et al. “Ten Years of Pedestrian Detection, What Have We Learned?”. In: *CoRR* abs/1411.4304 (2014). URL: <http://arxiv.org/abs/1411.4304>.
- [6] B. Benfold and I. Reid. “Stable multi-target tracking in real-time surveillance video”. In: *CVPR 2011*. 2011, pp. 3457–3464.
- [7] Ben Benfold and Ian Reid. “Guiding Visual Surveillance by Tracking Human Attention”. In: *Proceedings of the 20th British Machine Vision Conference*. 2009.
- [8] *Big O notation*. 2015. URL: https://en.wikipedia.org/wiki/Big_O_notation.
- [9] L. Bourdev, S. Maji, and J. Malik. “Describing people: A poselet-based approach to attribute classification”. In: *2011 International Conference on Computer Vision*. 2011, pp. 1543–1550. URL: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/poselets/attributes_dataset.tgz.

- [10] C. Chen, A. Heili, and J. M. Odobez. “A joint estimation of head and body orientation cues in surveillance video”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. 2011, pp. 860–867. DOI: [10.1109/ICCVW.2011.6130342](https://doi.org/10.1109/ICCVW.2011.6130342).
- [11] W. Choi and S. Savarese. “A Unified Framework for Multi-Target Tracking and Collective Activity Recognition”. In: *ECCV*. 2012.
- [12] W. Choi and S. Savarese. “A Unified Framework for Multi-Target Tracking and Collective Activity Recognition”. In: *ECCV*. 2012. URL: http://www-personal.umich.edu/~wgchoi/eccv12/wongun_eccv12.html.
- [13] Wongun Choi, Khuram Shahid, and Silvio Savarese. “Learning context for collective activity recognition”. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 3273–3280.
- [14] Wongun Choi, Khuram Shahid, and Silvio Savarese. “What are they doing?: Collective activity classification using spatio-temporal relationship among people”. In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 1282–1289.
- [15] François Chollet. *Keras*. 2015. URL: <https://github.com/fchollet/keras>.
- [16] Marco Cristani et al. “Social interaction discovery by statistical analysis of F-formations”. In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 23.1–23.12. ISBN: 1-901725-43-X. URL: <http://dx.doi.org/10.5244/C.25.23>.
- [17] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: [10.1109/CVPR.2005.177](https://doi.org/10.1109/CVPR.2005.177).
- [18] Zhiwei Deng et al. “Deep structured models for group activity recognition”. In: *arXiv preprint arXiv:1506.04191* (2015).
- [19] Zhiwei Deng et al. “Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4772–4781.
- [20] Piotr Dollár. *Piotr’s Computer Vision Matlab Toolbox (PMT)*. <https://github.com/pdollar/toolbox>.
- [21] Piotr Dollár et al. “Pedestrian Detection: An Evaluation of the State of the Art”. In: *PAMI* 34 (2012).

- [22] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *Kdd*. Vol. 96. 34. 1996, pp. 226–231.
- [23] W. Ge, R. T. Collins, and R. B. Ruback. “Vision-Based Analysis of Small Groups in Pedestrian Crowds”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.5 (2012), pp. 1003–1016. ISSN: 0162-8828. DOI: [10.1109/TPAMI.2011.176](https://doi.org/10.1109/TPAMI.2011.176).
- [24] Hossein Hajimirsadeghi et al. “Visual recognition by counting instances: A multi-instance cardinality potential kernel”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2596–2605.
- [25] Hossein Hajimirsadeghi et al. “Visual recognition by counting instances: A multi-instance cardinality potential kernel”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2596–2605.
- [26] David Hall and Pietro Perona. “Fine-Grained Classification of Pedestrians in Video: Benchmark and State of the Art”. In: *CoRR* abs/1605.06177 (2016). URL: <http://www.vision.caltech.edu/~dhall/projects/CRP/#dataset>.
- [27] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). URL: <http://arxiv.org/abs/1512.03385>.
- [28] Kurt Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. In: *Neural Netw.* 4.2 (Mar. 1991), pp. 251–257. ISSN: 0893-6080. DOI: [10.1016/0893-6080\(91\)90009-T](https://doi.org/10.1016/0893-6080(91)90009-T). URL: [http://dx.doi.org/10.1016/0893-6080\(91\)90009-T](http://dx.doi.org/10.1016/0893-6080(91)90009-T).
- [29] Jan Hendrik Hosang et al. “Taking a Deeper Look at Pedestrians”. In: *CoRR* abs/1501.05790 (2015). URL: <http://arxiv.org/abs/1501.05790>.
- [30] Mostafa S Ibrahim et al. “A hierarchical deep temporal model for group activity recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1971–1980.
- [31] Ke Jin, Peng Bao, and Weiwei Xing. “Novel Group Detection and Analysis Method Based on Automatic and Fast Density Clustering”. In: *DMS*. 2016.
- [32] Adam Kendon. *Conducting Interaction: Patterns of behavior in focused encounters*. Cambridge University Press, 1990. ISBN: 0-521-38036-7.
- [33] Saad M Khan and Mubarak Shah. “Detecting group activities using rigidity of formation”. In: *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM. 2005, pp. 403–406.
- [34] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). URL: <http://arxiv.org/abs/1412.6980>.

- [35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.
- [36] Tian Lan et al. “Beyond actions: Discriminative models for contextual group activities”. In: *Advances in neural information processing systems*. 2010, pp. 1216–1224.
- [37] A. Launila and J. Sullivan. “Contextual Features for Head Pose Estimation in Football Games”. In: *2010 20th International Conference on Pattern Recognition*. 2010, pp. 340–343. DOI: [10.1109/ICPR.2010.92](https://doi.org/10.1109/ICPR.2010.92).
- [38] Ruonan Li, Rama Chellappa, and Shaohua Kevin Zhou. “Learning multi-modal densities on discriminative temporal interaction manifold for group activity recognition”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 2450–2457.
- [39] Stephen J. McKenna et al. “Tracking groups of people”. In: *Computer Vision and Image Understanding* 80.1 (2000), pp. 42–56. ISSN: 1077-3142. DOI: [10.1006/cviu.2000.0870](https://doi.org/10.1006/cviu.2000.0870).
- [40] Alvis Memo, Ludovico Minto, and Pietro Zanuttigh. “Exploiting Silhouette Descriptors and Synthetic Data for Hand Gesture Recognition.” In: *Eurographics Italian Chapter Conference*. Ed. by Andrea Giachetti, Silvia Biasotti, and Marco Tarini. Eurographics Association, 2015, pp. 15–23. ISBN: 978-3-905674-97-2. URL: <http://dblp.uni-trier.de/db/conf/egItaly/egItaly2015.html#MemoMZ15>.
- [41] Anton Milan et al. “MOT16: A Benchmark for Multi-Object Tracking”. In: *CoRR* abs/1603.00831 (2016). URL: <http://arxiv.org/abs/1603.00831>.
- [42] C. Olah. *Understanding LSTM Networks*. 2015. URL: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [43] Javier Orozco, Shaogang Gong, and Tao Xiang. “Head Pose Classification in Crowded Scenes.” In: *BMVC*. British Machine Vision Association, 2009. URL: <http://dblp.uni-trier.de/db/conf/bmvc/bmvc2009.html#OrozcoGX09>.
- [44] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. “Improving Data Association by Joint Modeling of Pedestrian Trajectories and Groupings”. In: *Computer Vision – ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part I*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010,

- pp. 452–465. ISBN: 978-3-642-15549-9. DOI: [10.1007/978-3-642-15549-9_33](https://doi.org/10.1007/978-3-642-15549-9_33). URL: http://dx.doi.org/10.1007/978-3-642-15549-9_33.
- [45] William M. Rand. “Objective Criteria for the Evaluation of Clustering Methods”. In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. ISSN: 01621459. URL: <http://www.jstor.org/stable/2284239>.
- [46] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). URL: <http://arxiv.org/abs/1506.01497>.
- [47] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: [10.1007/s11263-015-0816-y](https://doi.org/10.1007/s11263-015-0816-y).
- [48] M. S. Ryoo and J. K. Aggarwal. *UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA)*. 2010. URL: http://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html.
- [49] MS Ryoo and JK Aggarwal. “Stochastic representation and recognition of high-level group activities”. In: *International journal of computer Vision* 93.2 (2011), pp. 183–200.
- [50] Michael S Ryoo and Jake K Aggarwal. “Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities”. In: *Computer vision, 2009 ieee 12th international conference on*. IEEE. 2009, pp. 1593–1600.
- [51] J. Shao, C. C. Loy, and X. Wang. “Scene-Independent Group Profiling in Crowd”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2227–2234. DOI: [10.1109/CVPR.2014.285](https://doi.org/10.1109/CVPR.2014.285).
- [52] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014).
- [53] F. Solera, S. Calderara, and R. Cucchiara. “Socially Constrained Structural Learning for Groups Detection in Crowd”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.5 (2016), pp. 995–1008. ISSN: 0162-8828. DOI: [10.1109/TPAMI.2015.2470658](https://doi.org/10.1109/TPAMI.2015.2470658).
- [54] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. “UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild”. In: *CoRR* abs/1212.0402 (2012). URL: <http://dblp.uni-trier.de/db/journals/corr/corr1212.html#abs-1212-0402>.
- [55] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting.” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

- [56] Patrick Sudowe, Hannah Spitzer, and Bastian Leibe. “Person Attribute Recognition with a Jointly-trained Holistic CNN Model”. In: *ICCV’15 ChaLearn Looking at People Workshop*. 2015. URL: <http://www.vision.rwth-aachen.de/page/parse27k>.
- [57] M. Voit, K. Nickel, and R. Stiefelhagen. “Multi-view head pose estimation using neural networks”. In: *The 2nd Canadian Conference on Computer and Robot Vision (CRV’05)*. 2005, pp. 347–352. DOI: [10.1109/CRV.2005.55](https://doi.org/10.1109/CRV.2005.55).
- [58] Mingrui Wu and Prof. Bernhard Schölkopf. “A Local Learning Approach for Clustering”. In: *Advances in Neural Information Processing Systems 19*. Ed. by P. B. Schölkopf, J. C. Platt, and T. Hoffman. MIT Press, 2007, pp. 1529–1536. URL: <http://papers.nips.cc/paper/3115-a-local-learning-approach-for-clustering.pdf>.
- [59] K. Yamaguchi et al. “Who are you with and where are you going?” In: *CVPR 2011*. 2011, pp. 1345–1352.
- [60] M. Zanotto et al. “Vision-Based Analysis of Small Groups in Pedestrian Crowds”. In: *British Machine Vision Conference*. 2012, 111.1–111.12.
- [61] Larry Zitnick and Piotr Dollar. “Edge Boxes: Locating Object Proposals from Edges”. In: *ECCV*. European Conference on Computer Vision, 2014. URL: <https://www.microsoft.com/en-us/research/publication/edge-boxes-locating-object-proposals-from-edges/>.