Deep Networks for Analyzing Group Activity

by

Ashish Goyal

in the supervision of

Prof. Rajbabu Velmurugan

Submitted in partial fulfillment for the degree of
Master in Technology

Department of Electrical Engineering
Indian Institute of Technology, Bombay
India

June 15, 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 [1]. We propose an objective function which learns amount of pairwise interaction (compatibility) between any two persons in a scene. We also train our own state of the art pedestrian pose (orientation) detector. As a minor contribution, we suggest post-processing steps to improve pedestrian detection for static cameras.

We evaluated our approach on standard datasets for group and scene activity, pose estimation and pedestrian detection. The results of scene activity 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 pose detector outperforms existing approaches. We also observe improvement in pedestrian detector performance due to the suggested post-processing algorithm. (To add: Does adding an extra step of computing group activities really help in scene activity? Does this approach help in surveillance?)

[1] Deep hierarchical framework for group and scene activity recognition was developed in collaboration with Neha Bhargava. Neha is associated with Vision and Image Processing Lab, Department of Electrical, IIT Bombay.

Dedication

To mum and dad

Declaration

I declare that..

Acknowledgments

I want to thank...

Contents

List of Figures

List of Tables

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 - $\boldsymbol{a}, \boldsymbol{A}, \boldsymbol{\Delta}$ 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, set of all real numbers is represented by $\mathbb{R}$.
- $\mathbb{R}^{a \times b \times c \times \cdots}$ is a $a \times b \times c \times ...$ dimensional space of real numbers.
- $A \in \mathbb{R}^{a \times b \times c \times \cdots}$ indicates A is a $a \times b \times c \times ...$ 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) [1], 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 etc
- Not involving humans - for e.g. roads, 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):

(a) Gesture [37]

(b) Action [48]

(c) Pairwise interaction [43]

(d) Group activity [9]

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

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

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, presence of objects like cars to list a few. Before going any further, it's critical to mention the types of videos we analyzed.

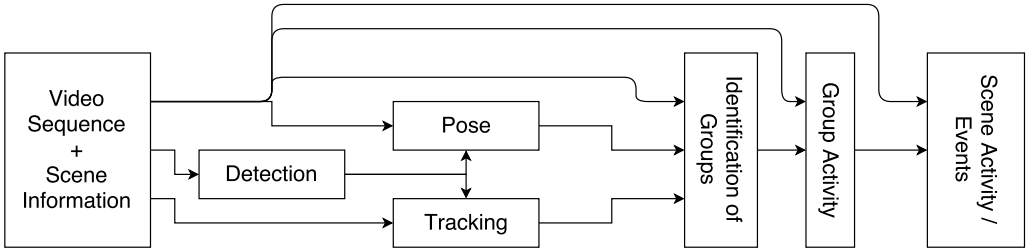


Figure 1.2: A flowchart depicting steps involved in activity recognition and event detection. Except for tracking, we address each step mentioned in this figure.

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 720x640 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 individual activity, person to person interaction, group discovery, recognition of group and scene activity, anomaly detection, personal space violation etc.

The next chapter explains various steps mentioned in Figure 1.2.

Chapter 2

Literature Survey

2.1 Person Detection

An object detector finds instances of objects in an image or video, for e.g. street signs, cars, persons, faces, road etc. Object detectors are generally based on a 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.

2.1.1 Region Proposals

Typically, patches within an image (also called region proposals) are fed to a classifier which detects the presence of a person (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 [18] for details). While this method is slow as it a generates 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 [54] or a neural network [41].

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) [27].

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

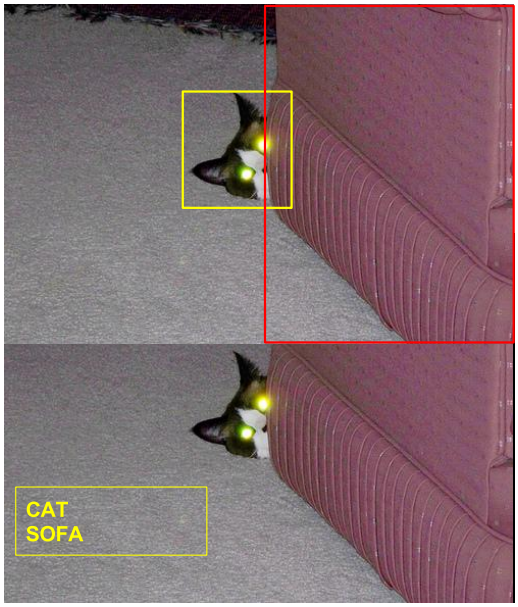


Figure 2.1: The difference between classifiers and detectors. The image on the top depicts detection results and the image on the bottom shows classification result.

refer to an excellent survey by Benenson *et al.* [4]. 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 CNNs are generally slow, a weak classifier is sometimes used to generate region proposals.

2.1.3 Datasets

Various datasets and benchmarks have been formed to organize the research in this area and evaluate performance of the detectors. Caltech pedestrian dataset [19] introduced in 2012, contains sequence of images containing pedestrians taken from a vehicle. INRIA person dataset [15] is a collection of images rather than a sequence.

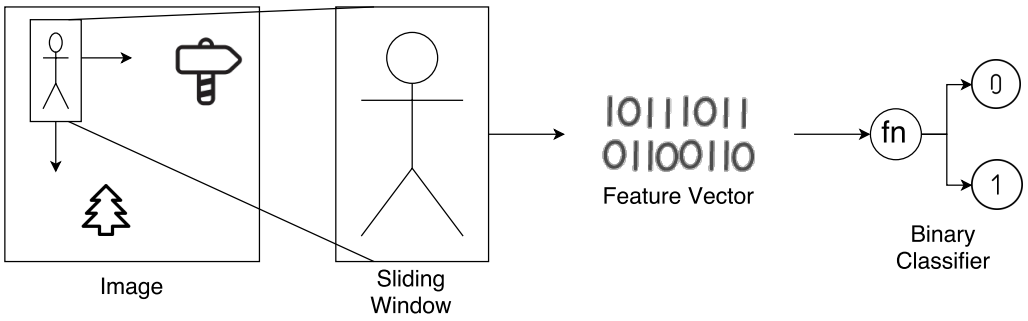
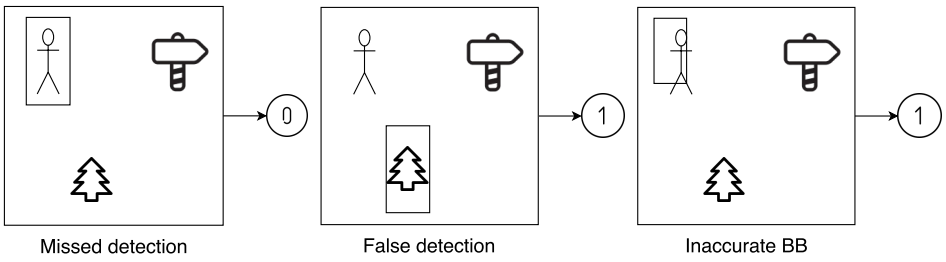


Figure 2.2: An overview of person detection with sliding window region proposals.



(a) (b) (c)

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.

2.1.4 Evaluation

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

- ***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:

$$Q = \frac{|B_{gt} \bigcap B_{det}|}{|B_{gt} \bigcup B_{det}|} \tag{2.1}$$

Where Q is the quality and $|\cdot|$ represents cardinality or area of the bounding box.

2.1.5 Challenges

To evaluate performance of person detectors in real surveillance videos, we analyzed a dataset obtained from Pune police in the context of detection using existing approaches [18, 4] (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 Pose Estimation

In this work, we define pose as the orientation of the human body in space. Intuitively, pose detection is critical to analyze interpersonal interactions and group formations. The Pose 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 pose estimation challenging and error-prone even for humans,

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

constraining data labeling. Thus, instead of treating pose as continuous quantity in $[0°, 360°]$, space is quantized in a number of bins (8 directions in this work). Under such constraints, pose 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 pose estimation too [35, 39, 6, 51]. To further improve performance, time filtering of the pose by combining information from velocity and appearance has been explored by Chen *et al.* [8].

Most approaches ignore the cyclic order in classes, thus, treating pose estimation simply as a classification problem. We train our own CNN with modified cross entropy loss, out-performing state of the art in terms of mean angular error.

2.2.1 Datasets

Pose of a person is one of its several attributes, thus annotated datasets for pose estimation generally are 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 pose detector. Parse27k [50] 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

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 [7]	8,035	17,628
Parse27k [50]	9887	$\sim$ 27,000
CRP [24]	20,999	27,454

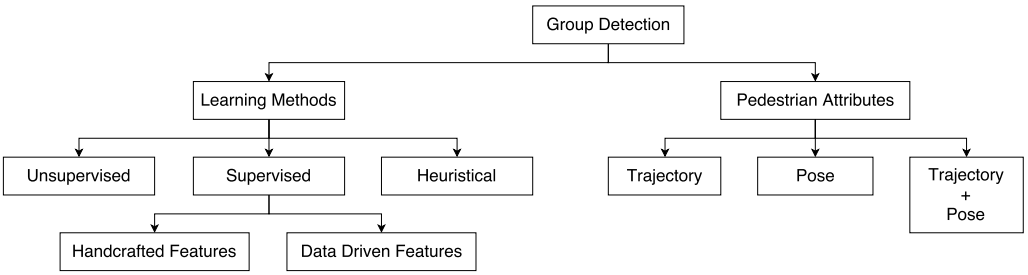


Figure 2.5: Division of various approaches for group detection. Our method uses data driven features and takes advantage of both trajectory and pose. It also learns the importance of each using training data, increasing its flexibility.

train our pose detector on this dataset.

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, most of the approaches depend upon the specific definition of a group and the choice of social reasoning. Figure 2.5 presents an overview to group detection methods, with detailed explanation in subsequent paragraphs.

2.3.1 Person Attributes

Benabbas *et al.* [3] track feature points using optical flow and use similarity of motion direction and spatial proximity to heuristically find groups. Jin *et al.* [29] define a pairwise affinity and use automatic fast density clustering to find group cores, which are further refined heuristically. Shao *et al.* [46] treat trajectories of individuals as a Markov chain and uses transition matrix to infer about groups. Pellegrini *et al.* [40] use a conditional random field to jointly estimate trajectories and group. While these approaches work well

in densely crowded dynamic scenes, they ignore pose 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 (pose) can be used to find groups. Bazzani *et al.* [2] introduce inter-relation pattern matrix to represent pairwise interaction. Marco *et al.* [14] estimate F-formation [30] using Hough-voting. However, pose estimates might not be reliable due to low resolution as mentioned in Section 1.2.1. Our approach uses pose 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 [40], classifiers like SVM [52] and/or heuristics [3, 29, 46, 14, 2]. 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.* 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 [21, 53], 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. 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 it from the training data itself.
- It only uses raw trajectories and pose 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 1. There are numerous approaches present in the literature on scene activity recognition. Some of the recent and interesting works are [10],[12],[11],[31],[36],[44],[45],[34],[22]. In [12], Choi *et al.* recognize the collective activity by extracting local spatio-temporal descriptors from people and the surroundings. They extend the algorithm in [11] by automatically capturing the crowd context and using it for classification. In [10], the authors go one step further and

present a unified framework for target tracking and activity recognition. Ryoo and Aggarwal in [44] model a group activity as a stochastic collection of individual activities. The method proposed in [22] is based on multi-instance cardinality model with hand crafted features.

Recently, deep learning based methods have also been explored to recognize activities [28, 17, 16, 23]. Deng *et al.* proposed a deep model [16] to capture individual actions, pairwise interactions, and group activity. In another work [17], 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 [28], 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.

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 pose estimation and group detection, a near complete void in literature for group activity recognition and scope of minor improvements in person 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.

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

Chapter 3

Methods

3.1 Person Detection

As discussed in 2.1.5, various challenges degrade the performance of person 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:

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

Here, $\{0, 1, ..., T\}$ 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:

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

(a) Background (I^b) (b) Frame (I^t) (c) Silhouette (I^s)

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.

Where $|.|$ is the absolute value and $I^s \in \mathbb{R}^{W \times H}$. 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:

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

This score $S_\mathcal{B}$ indicates weather $\mathcal{B}$ 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.

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 $\mathcal{B}$.

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:

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

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

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

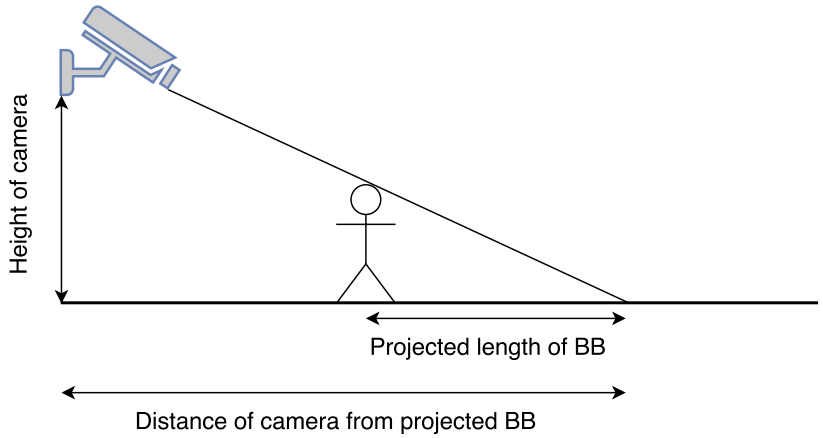


Figure 3.2: Calculation of scale factor $S(\mathcal{B})$ using projection image coordinates of $\mathcal{B}$ on to the ground plane.

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:

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

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 NMS except that we use total score $L_{\mathcal{B}}$ instead of classifier confidence $P_{\mathcal{B}}$. Note that $A \backslash B = \{x \in A | x \notin B\}$ denotes relative complement. $T_{ovlap} \in (0, 1)$ is the maximum amount of overlap (intersection over union) bounding boxes of two unique detections can have. Large value of T_{ovlap} introduces multiple detection for same person while small values tend to miss persons standing close to each other. Intersection over union (or IoU) is defined as $|\mathcal{B}_i \bigcap \mathcal{B}_j| / |\mathcal{B}_i \bigcup \mathcal{B}_j|$.

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).

Algorithm 1 Non-Maximum Suppression

procedure NMS
    input: $\mathcal{S} = \{(\mathcal{B}_0, L_{\mathcal{B}_0}), (\mathcal{B}_1, L_{\mathcal{B}_1}), ..., (\mathcal{B}_N, L_{\mathcal{B}_N})\}$,
top:
    if $|\mathcal{B}_i \bigcap \mathcal{B}_j| / |\mathcal{B}_i \bigcup \mathcal{B}_j| < T_{ovlap}$ for $i \neq j$, $\mathcal{B}_i, \mathcal{B}_j \in \mathcal{S}$ **then return** $\mathcal{S}$
    else goto *loop*
loop:
    counter = 0
    if $i \neq j$ **then**
        if $|\mathcal{B}_i \bigcap \mathcal{B}_j| / |\mathcal{B}_i \bigcup \mathcal{B}_j| > T_{ovlap}$ **then**
            if $L_{\mathcal{B}_i} < L_{\mathcal{B}_j}$ **then**
                $\mathcal{S} = \mathcal{S} \backslash (\mathcal{B}_i, L_{\mathcal{B}_i})$
                counter = counter + 1
            else
                $\mathcal{S} = \mathcal{S} \backslash (\mathcal{B}_j, L_{\mathcal{B}_j})$
                counter = counter + 1
    if counter > 0 **then goto** loop
    else return BB

Table 3.1: Table assigning poses to angles.

Label	Pose	Angle θ (in °)
1	Right	0
2	Right front	45
3	Front	90
4	Left front	135
5	Left	180
6	Left back	225
7	Back	275
8	Right back	315

3.2 Pose Estimation

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

We used ResNet50 [25] pretrained on ImageNet dataset [42] as our choice of CNN. ResNet50 (50 layers) has fewer number of parameters than its deeper counterparts having up to 152 layers. ResNet50 ($<$ 1 million parameters) has far lesser number of parameters than VGG16 ($>$ 130 million) [47]. 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 poses $[0°, 360°)$ 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:

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

Here, y_i is the predicted pose and $\hat{y}_i$ is the actual pose. $x\,rem\,y$ denotes remainder when x is divided by y. N is the number of data points. θ is a table (see Table 3.1) which assigns angle to each 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 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:

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

Here $\delta(x,y) = 1$ iff $x = y$ and 0 otherwise. $\mathcal{C}$ is the set of all poses (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 $\boldsymbol{\Delta_i} = [\delta(1,\hat{y}_i),\delta(2,\hat{y}_i),...,\delta(8,\hat{y}_i)]^T$, 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 $\mathbf{P(x_i)} = [log(P_\Theta(1|x_i)), log(P_\Theta(2|x_i)), ..., log(P_\Theta(8|x_i))]^T$ and $\boldsymbol{\bar{P}(x_i)} = [log(1-P_\Theta(1|x_i)), log(1-P_\Theta(2|x_i)), ..., log(1-P_\Theta(8|x_i))]^T$, 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:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\Delta_i}^T\cdot\mathbf{P(x_i)} \tag{3.8}$$

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

$$K = -\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\Delta_i}^T\boldsymbol{D}\cdot\boldsymbol{\bar{P}(x_i)} \tag{3.9}$$

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

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

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

$$J = -\left(w_1 \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\Delta}_{\boldsymbol{i}}^{T} \cdot \mathbf{P}(\mathbf{x_i}) + w_2 \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\Delta}_{\boldsymbol{i}}^{T} \boldsymbol{D} \cdot \bar{\boldsymbol{P}}(\boldsymbol{x_i}) \right) \tag{3.10}$$

```
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, input_shape=(224, 224, 3))
6  input_layer = Input(shape=(224, 224, 3))
7  resnet_features = resnet(input_layer)
8  resnet_features = Reshape(target_shape=(2048, ))(
       resnet_features)
9  resnet_dense = Dense(1024, activation='relu')(
       resnet_features)
10 resnet_prob = Dense(8, activation='softmax')(
       resnet_dense)
11 pose_resnet = Model(input=input_layer, output=
       resnet_prob)
```

Listing 3.1: Keras [13] code to define the neural network architecture for pose estimation. `include_top=False` in 5^{th} line removes top two clayers of the ResNet50. This code allows training of all layers.

3.3 Group Detection [1]

We pose group detection as a supervised clustering problem. Consider a set of n people $\mathcal{P} = \{\boldsymbol{p_1}, \boldsymbol{p_2}, ..., \boldsymbol{p_n}\}$. A clustering problem is to find K partitioning $\mathcal{C} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_K\}$, $\mathcal{G}_i \subseteq \mathcal{P}\ \ \forall\, i \in \{1, 2, ..., K\}$ such that:

$$\mathcal{G}_i \bigcap \mathcal{G}_j = \phi \quad \forall\, i \neq j$$
$$\bigcup_{i=1}^{K} \mathcal{G}_i = \mathcal{P}$$

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 $\mathcal{T} = \{(\mathcal{P}^1, \mathcal{C}^1), (\mathcal{P}^2, \mathcal{C}^2), ..., (\mathcal{P}^N, \mathcal{C}^N)\}$ 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 $\boldsymbol{u} \in \mathcal{P}$, where $\mathcal{P}$ has a known partitioning $\mathcal{C} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_K\}$. Also let $\mathcal{G}(\boldsymbol{u})$ denote the group containing $\boldsymbol{u}$, i.e. $\mathcal{G}(\boldsymbol{u}) = \mathcal{A} \in \mathcal{C} | \boldsymbol{u} \in \mathcal{A}$. We define:

$$\alpha_{\boldsymbol{u}} = \max_{\boldsymbol{v}\,|\,\boldsymbol{v}\neq\boldsymbol{u},\ \boldsymbol{v}\in\mathcal{G}(\boldsymbol{u})} f(\boldsymbol{u}, \boldsymbol{v})$$
$$\beta_{\boldsymbol{u}} = \max_{\boldsymbol{v}\,|\,\boldsymbol{v}\neq\boldsymbol{u},\ \boldsymbol{v}\notin\mathcal{G}(\boldsymbol{u})} f(\boldsymbol{u}, \boldsymbol{v})$$

Here $\alpha_{\boldsymbol{u}}$ is the maximum possible intra-group affinity and $\beta_{\boldsymbol{u}}$ is the maximum possible inter-group affinity for element $\boldsymbol{u}$. It is desirable to maximize $\alpha_{\boldsymbol{u}}$ and minimize $\beta_{\boldsymbol{u}}$ for each individual $\boldsymbol{u} \in \mathcal{P}$. Hence, we minimize an

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

objective function J^i for i^{th} training tuple $(\mathcal{P}^i, \mathcal{C}^i)$ as:

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

A shown by Hornik [26] 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 $f : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$. 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 3.11 using gradient descent as it is differentiable almost everywhere in its domain.

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:

$$tanh(z) = \frac{(e^{(2z)} - 1)}{(e^{(2z)} + 1)}$$
$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

```
from keras.models import Model
from keras.layers import Input, Dense

n_hidden1 = 2**5
n_hidden2 = 2**5
f_len = 18

in1 = Input(shape=(f_len*2, ))
fc1 = Dense(n_hidden1, activation='tanh')(in1)
fc2 = Dense(n_hidden2, activation='tanh')(fc1)
out1 = Dense(1, activation='sigmoid')(fc2)
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 pose.

Upon the completion of the optimization problem in Equation 3.11, the

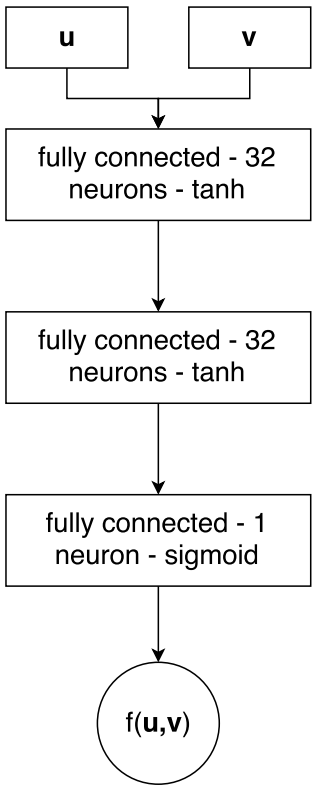
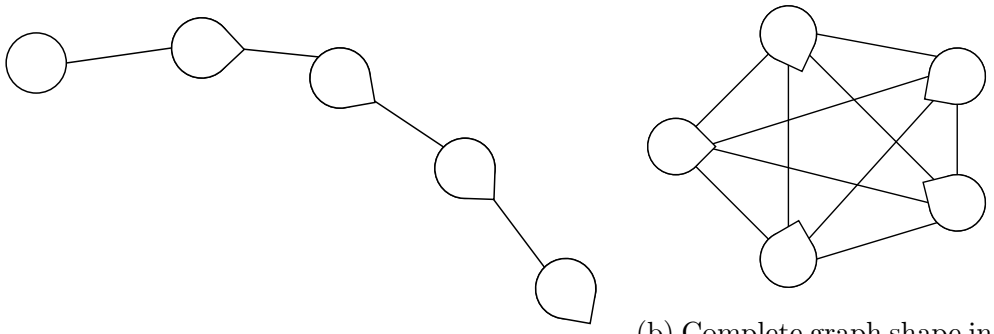


Figure 3.3: A simple multilayer feedforward neural network architecture for learning affinity function.

function f_w can be used to predict affinity between any two persons $\boldsymbol{u}$ and $\boldsymbol{v}$. Suppose there are N people $\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_N \in \mathcal{P}$, where $\mathcal{P}$ is a new set to be clustered. We compute the $N \times N$ affinity matrix $\boldsymbol{A}$ such that $\boldsymbol{A_{ij}} = f_w(\boldsymbol{u}_i, \boldsymbol{u}_j)\ \forall\ i, j \in \{1, 2, ..., N\}$. Clustering is performed on the affinity matrix $\boldsymbol{A}$ using DBSCAN [20] algorithm. This algorithm has following properties essential for our task (see [20] 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.



(a) Linear chain shape in queue groups.

(b) Complete graph shape in talking groups.

Figure 3.4: Examples of different shaped groups possible in surveillance videos. DBSCAN can discover these clusters based on pairwise affinities.

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 $\boldsymbol{f}_w$ between two individuals. During testing, we use this measure to find the affinity matrix $\boldsymbol{A}$, which is used by DBSACN algorithm to find the groups. Learning affinity measure $\boldsymbol{f}_w$ becomes non-trivial when pairwise interactions are not available. Therefore, we formulated an optimization problem 3.11.

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. Evident from examples, such rich interaction information will definitely be extremely helpful for activity analysis. However, there are N^2 pairs to be annotated for N individuals, increasing the labeling effort multiple-folds.

Choi *et al.* [9] 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 [2]

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, pose and individual action information directly to recognize the group's activity. Consider a group $\mathcal{G} = \{\boldsymbol{p}_1, \boldsymbol{p}_1, ..., \boldsymbol{p}_N\}$ of size N where $\boldsymbol{p}_1, \boldsymbol{p}_1, ..., \boldsymbol{p}_N$ are feature vectors of N individuals, consisting of their location, velocity, pose 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 $\boldsymbol{g} = [\boldsymbol{p}_1 \frown \boldsymbol{p}_1 \frown ... \frown \boldsymbol{p}_N]$, where $[\boldsymbol{x} \frown \boldsymbol{y}]$ is concatenation of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Note that $dim(\boldsymbol{g}) = K \cdot dim(\boldsymbol{p}_1)$.
- $N < K$: Concatenate features of each individual forming a vector $\boldsymbol{g} = [\boldsymbol{p}_1 \frown \boldsymbol{p}_1 \frown ... \frown \boldsymbol{p}_N]$. Since $dim(\boldsymbol{g}) < K \cdot dim(\boldsymbol{p}_1)$, pad the vector $\boldsymbol{g}$ with zeros to make $dim(\boldsymbol{g}) = K \cdot dim(\boldsymbol{p}_1)$.
- $N > K$: 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 $\boldsymbol{g}$ of dimension $dim(\boldsymbol{g}) = K \cdot dim(\boldsymbol{p}_1)$.

We eliminate the need of handcrafted features by directly feeding vector $\boldsymbol{g}$ 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 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 $\boldsymbol{g}$ for 10 frames. State of the LSTM at every 10^{th} 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.

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

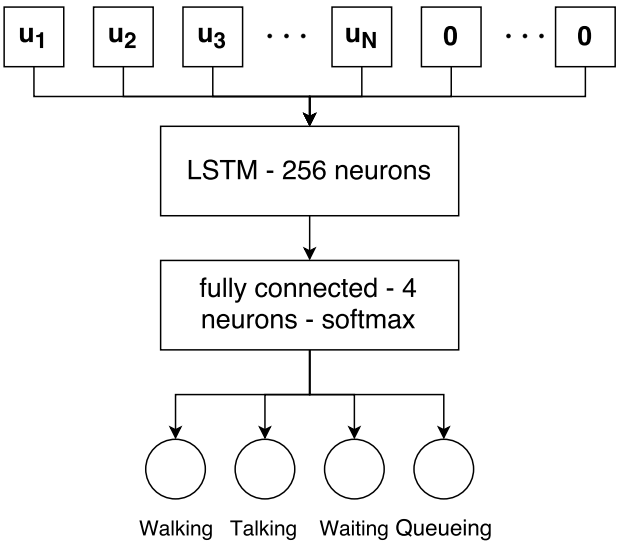


Figure 3.5: Neural network architecture for group detection. We use 10 frames as our sequence length, primarily because annotations were provided at those intervals. Since activities are simple, observing sequences for longer durations does not help.

```
from keras.models import Model
from keras.layers import Input, LSTM, Dense,
   Masking

input_layer = Input(shape=(10, 180))
masked_in = Masking()(input_layer)
lstm1 = LSTM(256, activation='sigmoid',
   recurrent_activation='tanh',
return_sequences=False)(masked_in)
fc1 = Dense(4, activation='softmax')(lstm1)
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 lnegth less than 10.

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

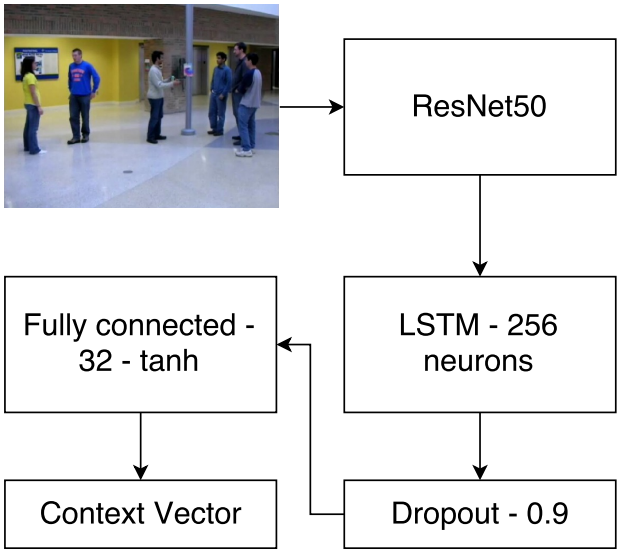


Figure 3.6: Neural network architecture for context unit. Pretrained ResNet50 is used as described in Section 3.2 with fixed weights.

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 [49] 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.

```
from keras.models import Model
from keras.layers import Input, LSTM, Dense,
    Masking, Dropout

resnet_layer = Input(shape=(10, 2048))
masked = Masking()(resnet_layer)
lstm1 = LSTM(256, activation='sigmoid',
    recurrent_activation='tanh')(masked)
drop1 = Dropout(rate=0.9)(lstm1)
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.

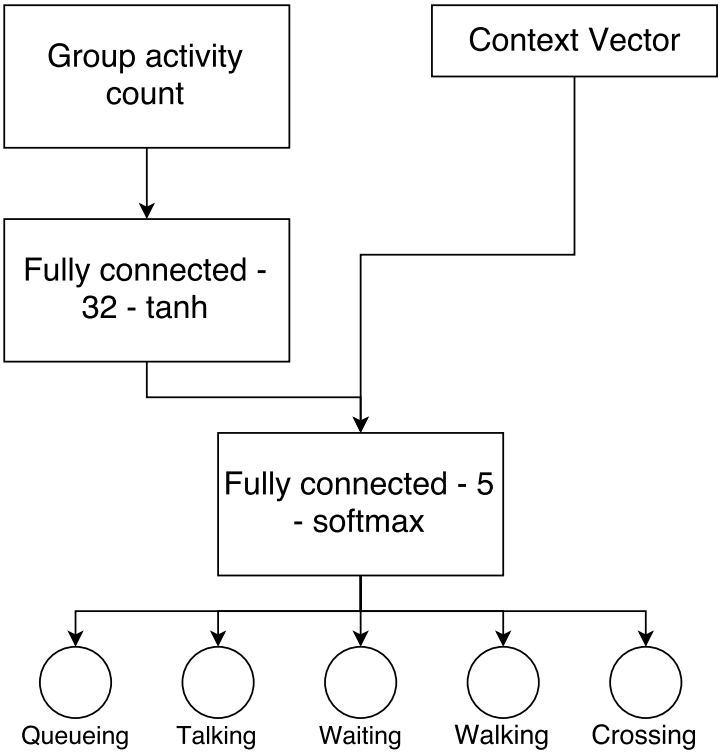


Figure 3.7: Neural network for recognizing scene activity from groups.

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 propagate.

Scene activity from individuals: As in group analysis 3.4.1, we concatenate individual descriptors (location, velocity, pose 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).

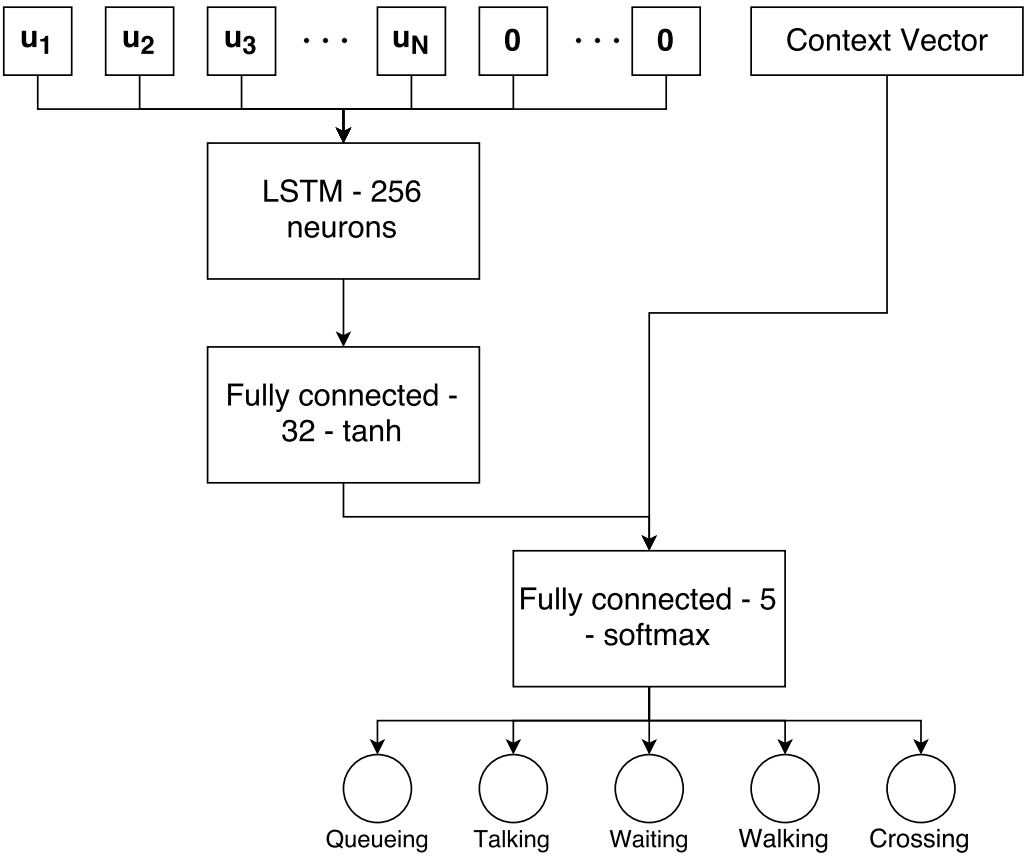


Figure 3.8: Neural network for recognizing scene activity from individuals.

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.

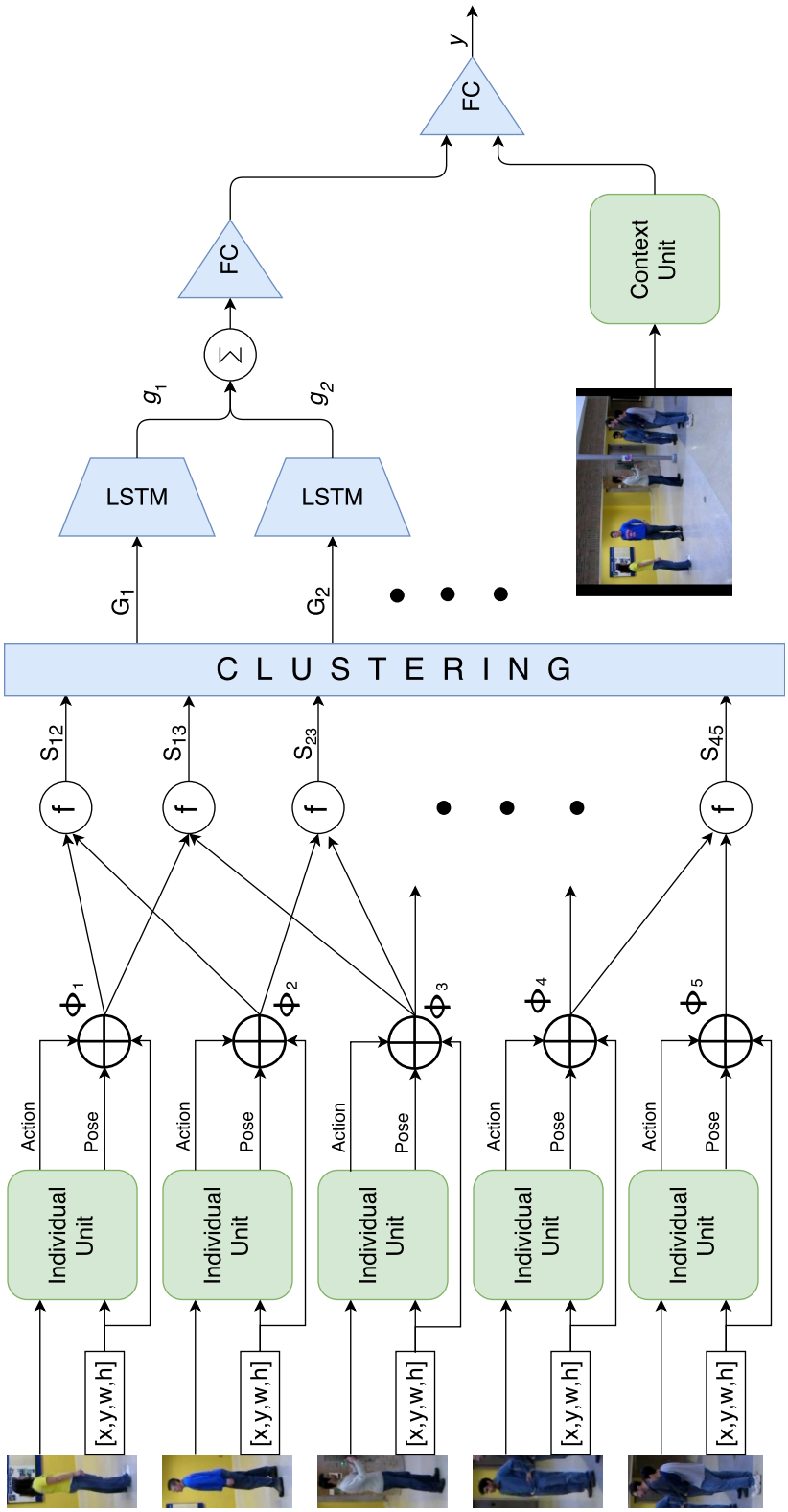


Figure 3.9: Summary of the complete model with various levels of hierarchy.

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:

$$Q = \frac{|B_{gt} \bigcap B_{det}|}{|B_{gt} \bigcup B_{det}|} \tag{4.1}$$

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 [38] and Towncentre [5], since ground truth is available and 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 [38] and Towncentre [5] sequences. Our post-processing steps clearly indicate improvements over baseline [18]. 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

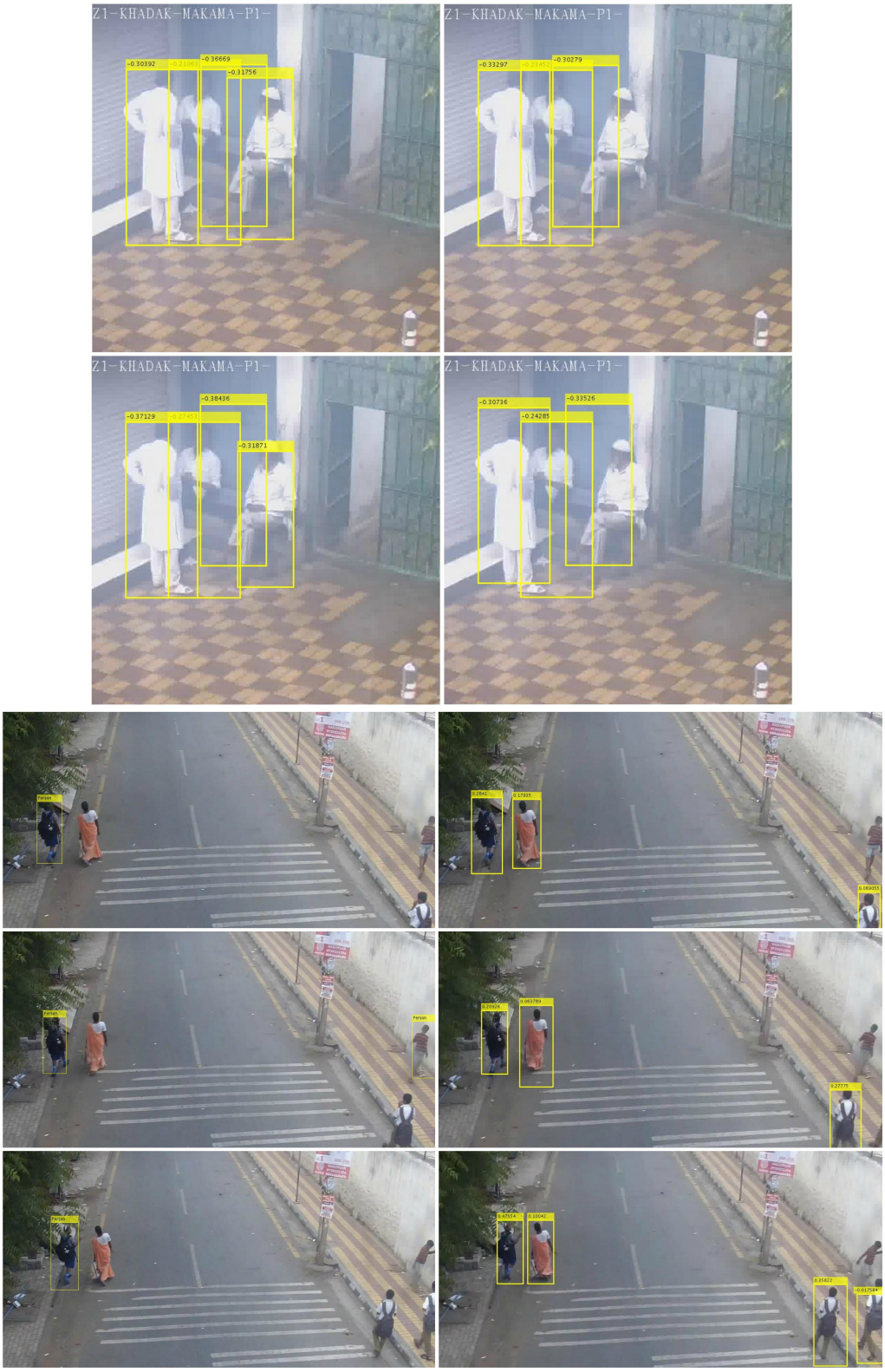


Figure 4.1: Qualitative comparison of baseline [18] before and after and post-processing steps. Left column contains the results before post-processing while the right column contains the results after post-processing.

4.2 Orientation Estimation

We trained and evaluated our approach for orientation estimation on Parse27k [50] dataset described in Section 2.2. Figure 4.2 depicts selected images from aforementioned dataset. We also compare variations of our approach as tabulate in Table 4.2.

Table 4.2: Three variations of the described model were investigated. This table describes the differences among the three models.

Model Name	Desciption
total_loss_all_layers	Instead of standard cross-entropy loss, we used the objective defined in Equation 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 to train the network. Only last two fully-connected (fc) layers were finetuned.
standard_loss_fc_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 [13] implementation[1] by subtracting mean $[103.939, 116.779, 123.68]$ for RGB channels respectively. These values are channel-wise mean of ImageNet ILSVRC15 [42] training set images. We used a batch of 32 randomly sampled images from the dataset and fixed a learning rate of 0.0001. We used Adam [32] optimizer, eliminating the need of reducing learning rate since it adapts automatically. We train the model for 10 epochs, each epoch takes $\sim$ 10 minutes on NVIDIA GTX 1080 GPU.

We compare our results with [50], where Parse27k dataset was originally proposed. We take their best performing model - Attributes Convolutional Net with hidden layers for each attribute (ACNH) trained separately for eight orientation estimation with data augmentation (random cropping, mirroring, and scaling) and PCA jittering as suggested in [33].

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

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

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

Figure 4.2: Selected frames from the Parse27k [50] dataset used for training and testing the network.

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

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

Table 4.5: Summary of results for orientation estimation including all the models we investigated.

Model Name	Accuracy (in %)	Angular Error
ACNH [50]	73.8	-
total_loss_fc_layers	60.1	

Bibliography

[1] 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.

[2] 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. URL: http://dx.doi.org/10.1007/978-3-642-28598-1_9.

[3] 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. URL: http://dx.doi.org/10.1155/2011/163682.

[4] 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.

[5] B. Benfold and I. Reid. "Stable multi-target tracking in real-time surveillance video". In: *CVPR 2011*. 2011, pp. 3457–3464. DOI: 10.1109/CVPR.2011.5995667.

[6] Ben Benfold and Ian Reid. "Guiding Visual Surveillance by Tracking Human Attention". In: *Proceedings of the 20th British Machine Vision Conference*. 2009.

[7] 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.

[8] 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.

[9] W. Choi and S. Savarese. "A Unified Framework for Multi-Target Tracking and Collective Activity Recognition". In: *ECCV*. 2012.

[10] 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.

[11] 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.

[12] 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.

[13] François Chollet. *Keras*. 2015.

[14] Marco Cristani et al. "Social interaction discovery by statistical analysis of F-formations". In: *Proceedings of the British Machine Vision Conference*. http://dx.doi.org/10.5244/C.25.23. BMVA Press, 2011, pp. 23.1–23.12. ISBN: 1-901725-43-X.

[15] 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.

[16] Zhiwei Deng et al. "Deep structured models for group activity recognition". In: *arXiv preprint arXiv:1506.04191* (2015).

[17] 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.

[18] Piotr Dollár. *Piotr's Computer Vision Matlab Toolbox (PMT)*. https://github.com/pdollar/toolbox.

[19] Piotr Dollár et al. "Pedestrian Detection: An Evaluation of the State of the Art". In: *PAMI* 34 (2012).

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). URL: http://arxiv.org/abs/1512.03385.

[26] 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. URL: http://dx.doi.org/10.1016/0893-6080(91)90009-T.

[27] Jan Hendrik Hosang et al. "Taking a Deeper Look at Pedestrians". In: *CoRR* abs/1501.05790 (2015). URL: http://arxiv.org/abs/1501.05790.

[28] 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.

[29] Ke Jin, Peng Bao, and Weiwei Xing. "Novel Group Detection and Analysis Method Based on Automatic and Fast Density Clustering". In: *DMS*. 2016.

[30] Adam Kendon. *Conducting Interaction: Patterns of behavior in focused encounters*. 1990.

[31] 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.

[32] 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.

[33] 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.

[34] Tian Lan et al. "Beyond actions: Discriminative models for contextual group activities". In: *Advances in neural information processing systems*. 2010, pp. 1216–1224.

[35] 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.

[36] 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.

[37] Alvise 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.

[38] Anton Milan et al. "MOT16: A Benchmark for Multi-Object Tracking". In: *CoRR* abs/1603.00831 (2016). URL: http://arxiv.org/abs/1603.00831.

[39] 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.

[40] 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. URL: http://dx.doi.org/10.1007/978-3-642-15549-9_33.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014).

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] K. Yamaguchi et al. "Who are you with and where are you going?" In: *CVPR 2011*. 2011, pp. 1345–1352.

[53] M. Zanotto et al. "Vision-Based Analysis of Small Groups in Pedestrian Crowds". In: 2012, 111.1–111.12.

[54] 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/.